exelvision

banc d'essai Exeltel-VS

les histogrammes

initiation assembleur : le VDP

LA REVUE EXELEMENT VOTRE

15

ISSN 0769-3826 - N° 15 PRIX - 25 F

EDITOMATIQUE

Chers Télémates,

Vous qui rêvez d'ajouter à votre EXELTEL la possibilité de transmettre plus qu'une simple page VIDEOTEX,
Vous qui osez imaginer que votre EXELTEL est capable de faire ce que font les serveurs les plus récents
Vous qui fantasmez sur les affichages dynamiques des stars de Télétel 3
Vous avez raison !

Tout ce que vous imaginiez sous basic et vous auriez souhaité réaliser sous vidéotex, vous allez pouvoir le programmer! Je ne vous en dis pas plus car nous en sommes seulement qu'à l'édito
Rendez-vous en page 2 pour approfondir ce qui est en fait l'EXELTEL-VS

NOTE Pour 890 F vous pouvez faire transformer votre EXELTEL en EXELTEL-VS par notre SAV! (Envoyez votre machine + chèque à EXELVISION VALBONNE)

Les adeptes d'EXELMAX trouveront un complément précieux pour manipuler les registres du VDP (Video Display Processor) L'initialisation de ce dernier ne devrait plus poser le moindre problème

Les fanatiques de la programmation basic n'auront plus à se préoccuper de savoir comment représenter graphiquement des données numériques Un véritable exposé, mieux, une thèse leur fera découvrir les secrets des histogrammes qui tiennent debout!

Ah, j'oubliais précipitez-vous sur le bulletin de réabonnement, car contrairement à ce que nous vous avions annoncé dans le numéro précédent, le système de réabonnement n'est pas encore automatique Beaucoup d'entre vous n'ont pas reçu le numéro 14 ou même le numéro 13 et nous ont légitimement téléphoné La grande majorité arrivait en fin d'abonnement

La question du mois:

Est-ce qu'EXELVISION abandonne le marché grand public pour s'orienter uniquement sur les marchés télématiques professionnels?

Nous avons eu la mauvaise surprise (autant que vous !) de lire dans les colonnes d'un mensuel français de la micro-informatique, que nous avions l'intention de faire une chose pareille, voire même que nous vous avions déjà abandonné, vous nos chers lecteurs!

Rassurez-vous, il n'en n'est rien!
D'abord pourquoi le ferions nous? Nous ne sommes pas un certain constructeur français bien connu!
Nous n'avons pas d'usine en France, donc aucune raison de la fermer! Comme quoi, dans les journaux qui veulent ou qui croient tout savoir, il traîne pas mal de suppositions douteuses qui bien souvent se trouvent transformées en affirmations

Cela revient presque à insinuer que nous n'avons pas le courage de faire à la fois du grand public et du professionnel ! Vous voulez savoir d'où vient la rumeur? C'est tout simplement parce que nous avons eu un hhennooooorrrmme retard dans l'installation des logiciels EXELQUAD sur le serveur Je dis nous avons eu car, vous vous en doutez, à présent on sort du tunnel: au moment où vous lirez ces lignes, le nombre de têtes de série de logiciels EXELQUAD disponibles en téléchargement aura nettement augmenté

A propos de marché grand public, il faut être rudement rétrograde (et fermé au progrès) pour penser que Vous, cher lecteur et client, n'êtes pas capable d'évoluer tout doucement vers le professionnel Et bien malin, celui qui, aujourd'hui peut prétendre définir exactement ce que veut ce fameux marché grand public!
On dirait que ce fameux mensuel micro-informatique considère que les marchés grand-public et professionnels sont complètement disjoints et que le grand public (Vous et moi) n'achètera jamais un ordinateur pour travailler ou même simplement se faciliter la vie! Etrange analyse quand on sait que le marché dit "grand public" est en train de s'orienter vers une multitude de micro-applications avant tout utiles et que chacun d'entre vous souhaite pouvoir maîtriser simplement

Tandis que certains constructeurs ont résolument choisi le marché de la P A O (publication assistée par ordinateur), Exelvision préfère le marché des terminaux (terminaux multi-normes, bi-standard, programmables, ANSI,) qui ,professionnel a priori, devient déjà grand public On ne peut d'ailleurs s'empêcher de faire le parallèle avec le marché des téléviseurs multistandard, uniquement professionnel ou frontalier à ses premières heures, aujourd'hui marché de référence

BANC D'ESSAI

EXELTEL-VS

EXELTEL ou EXELTEL-VS ?

L'EXELTEL-VS est la version professionnelle de l'EXELTEL Il fait partie de la famille des terminaux bi-standard ASCII-VIDEOTEX programmables

VS signifieVERSION STANDARD Celle-ci ne remplace en aucune manière l'EXELTEL éducatif qui a encore de longs jours devant lui

L' EXELTEL-VS est la base d'une série de machines prévues pour transmettre, recevoir et traiter des données dans n'importe quel contexte

L'EXELTEL-VS est également doté de fonctions bien pratiques et utilisables par tous (EXELTELEX, TELETEL) et peut recevoir la quasi-totalité des logiciels en cartouche prévus pour les machines précédentes

La partie cachée de l'iceberg consiste en une refonte du langage basic, qui, s'étant débarrassé de quelques instructions moins professionnelles, a reçu la possibilté d'afficher en 80 colonnes et surtout la possibilité de programmer le modem à 100% !

Faire son propre serveur VIDEOTEX relève à présent du simple exercice pour débutant en programmation basic! Quelques exemples de programmes sont d'ailleurs fournis dans le manuel d'utilisation. Nous en reparlerons plus loin

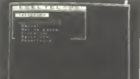
L'EXELTEL-VS comprend

- l'unité centrale
- le clavier
- le moniteur monochrome ou couleur
- l'Exelmémoire
- l'interface multi-fonctions
- les cordons de raccordement
- les manuels d'utilisation
- les cartes de garantie

Menu général de l'EXELTEL-VS



La fonction TELEPHONE



Comme sur l'EXELTEL, on retrouve les fonctions de numérotation et de rappel du dernier numéro

Les fonctions CONNEXION et RACCROCHE permettent respectivement la numérotation et une conversation préalables sur un équipement extérieur et la possibilité de raccrocher en cas d'appel aboutissant à un répondeur téléphonique.

Un contrôle par système de mot de passe permet d'autoriser ou non l'accès à une liste de numéros téléphoniques protégés

BANC D'ESSAI (SUITE)

EXELTEL-VS

TRAITEMENT DES PAGES VIDEOTEX ►

Comme sur l'EXELTEL, on retrouve le menu horizontal des gestion des pages mémorisées en cours de communication. Les sauvegardes et les chargements peuvent adresser aussi bien l'Exelmémoire que l'EXELDISK.

A noter l'apparition du mode d'impression graphique inversé permettant d'imprimer des pages minitel dont le fond est noir en conservant comme couleur de fond celle du papier à l'impression...





◄FONCTION EXELTELEX

L'EXELTEL-VS permet également la réception de messages suivant le principe du télex. N'importe quel terminal VIDEOTEX est capable de transmettre un message et d'avoir la confirmation que ce dernier a bien été reçu. L'utilisateur a la possibilité de personnaliser le message d'accueil grâce à une ligne programmable. L'émission d'un message à partir d'un EXELTEL-VS est évidemment possible. L'utilisateur peut alors saisir son texte en mode local et le transmettre à un ou plusieurs correspondants. Il est également possible d'envoyer des textes tapés avec le traitement de texte EXELTEXTE 80.

FONCTION CALCULATRICE ►

Cette fonction offre à l'utilisateur la possibilité de réaliser des calculs en mode direct ou en mode programmé. Il s'agit en fait du point d'entrée de l'EXELBASIC-VS qui se trouve donc intégré à la machine. Comme précisé dans le manuel d'utilisation, il est dérivé de l'EXELBASIC + et enrichi d'un ensemble de fonctions permettant (enfin!) de programmer le modem en basic. Le retour au menu général de l'EXELTEL-VS s'obtient en tapant EXIT. Le mode calculatrice permet à chacun d'aborder en souplesse les principes de la programmation qui, dans certains calculs permet de gagner un temps précieux.





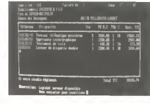
◄ TABLEAU DE BORD

Il permet de gérer efficacement les coulisses de l'EXELTEL ainsi que l'accès aux périphériques. Le répertoire, de taille variable, permet bien sûr la numérotation directe et fait l'objet d'un tri alphabétique. A chaque numéro téléphonique répertorié il est possible d'y juxtaposer le nom du service ainsi que les mnémoniques suivies d'éventuelles commandes (ENVOI,SUITE, etc...). Cette facilité s'appelle le séquencement automatique.

EXELTEL-VS

BANC D'ESSAI (SUITE)

L'EXELBASIC-VS...

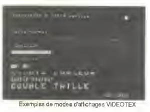


Encore un petit mot sur ce qui est en réalité bien plus qu'une simple calculatrice
Aprés avoir une subi une cure d'amaignssement (haute résolution graphique, rappel des "touches BASIC" et mémorisation des touches de fonctions en moins), l'EXELBASIC-VS a reçu en revanche des instructions bigrement intéressantes et bien sur l'affichage 80 colonnes monochrome

Nul doute que même les programmes basic qui n'ont rien a voir avec la télématique y gagneront en professionnalisme!
Notons tout de même que la vitesse d'affichage dans ce mode est moins rapide qu'en mode 40 colonnes couleur ce qui est normal Rassurez-vous, cela n'affecte en rien la vitesse de traitement. Le passege en 80 colonnes s'obtient grâce a l'instruction CALL VIDEO("H") et s'accompagne d'une réduction de la teille de la mémoire disponible. Le mode inverse s'obtient avec CALL VIDEO ("h") Un seul générateur de caractéres est utilisé, mais les caractéres semi-graphiques qu'il contient permettent de representer n'importe quel tableau ou encadré . Le retour au mode 40 colonnes se fait en tapant CALL HROFF



Page d'accueil d'un serveur réalisé sur EXELTEL-VS



Exemplss de modes d'affichages VIDEOTEX

Les commandes permettant de gérer le modem sont au nombre de onze. Avec l'EXELBASIC-VS, il devient aussi facile d'envoyer des caractéres á un Minitel vie le modem que de les éditer sur votre imprimante! La réception dee caractéres provenant du modem est permanente grâce á l'utilisation d'un buffer FIFO (first In is first out) de 255 caractéres

Celui-ci doit être vidé réguliérement par une instruction de type LINPUT, A$ La chaine alphanumérique A$ contient alors la totalité des caractéres reçus, que ce soit d'un terminal ou d'un serveur Le réversibilité et le choix des vitessses se gére par l'instruction DELETE "30.VD1" et DELETE "30 VD2". L'envoi de messages vocaux sur la ligne reste possible. Le passage en mode beille relève de l'instruction DELETE"30.Z"

Au réveil, le mode de travail est automatiquement re-sélectionné (merci á l'EEPROM !). Mieux: non seulement votre EXELTEL-VS revient en mode CALCULATRICE (ou si vous préferez sous EXELBASIC-VS), mais en plus il va voir s'il doit éxécuter un programme basic sauvegardé dans l'EXELMEMOIRE. Pour être éxécuté, ce programme doit s'appeler AUTOEXEC BAS

Voici un récapitulatif que tout programmeur branché VIDEOTEX a besoin de connaître



...la télématique encore plus facile à vivre !

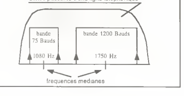
Ce qu il faut bien comprendre, c'est qu'un terminal Videotex et un serveur reliés ensemble forment un couple en pleine discussion Le serveur "parle" beaucoup plus vite (1200 Bauds) que le terminal (75 Bauds), ce qui est normal puis que c'est le serveur qui fournit les informations alors que le terminal se contente de manifester ses choix Malgre la différence de vitesses de communication, les deux partenaires se comprennent parfaitement même s'ils emettent des donnees simultanement en sens inverse Cela s'appelle communiquer en full duplex

Jusqu'à présent, tout est pour le mieux et tout semble transparent Profitons-en pour expliquer succintement le principe d'échanges de signaux de directions inverses sur une ligne telephonique apparemment mono-canal

La solution consiste à diviser en deux bandes actives la plage de fréquences utilisable par une ligne telephonique traditionnelle

Chaque bande est affectee à un sens de transfert d'informations Le systeme de codage est obtenu par la modulation des frequences médianes

La bande dont le debit est le plus important (1200 bd) est la plus large afin d'obtenir une qualité de transmission (rapport S/B) meilleure Le taux d'erreur reste alors acceptable



Comme deja décrit précedemment les caractères reçus vont être récupérés par l'instruction LINPUT , A$ La chaine de caractera A$ contient pelle-mêle la totalité des caractères émis En clair, cela signifie qu'il va falloir trier les caractères utiles, les caracteres de commandes ainsi que les séquences de contrôle qui servent à gérer l'affichage

Les caractères de commande

Les caractères de commande correspondent aux touches de commande que va taper l'utilisateur a partir de son terminal VIDEOTEX (RETOUR,SUITE, SOMMAIRE, etc) Ces commandes sont précédées du code 19 Ce qui donne

CHR$(19) puis A	ENVOI
CHR$(19) puis B	RETOUR
CHR$(19) puis C	REPETITION
CHR$(19) puis D	GUIDE
CHR$(19) puis E	ANNULATION
CHR$(19) puis F	SOMMAIRE
CHR$(19) puis G	CORRECTION
CHR$(19) puis H	SUITE
CHR$(19) puis I	CONNEXION/FIN

Il conviendra lors de la récupération d'une chaine de caractere, de tester si celle-ci contient une valeur ASCII egale a 19 Si une telle valeur est détectee, il suffit d'identifier quelle est la valeur suivante et de brancher le sous-programme correspondant au traitement de la commande



Les codes de fonctions

A l'inverse des caractères de commande èmis par tout terminal VIDEOTEX, les codes de fonction sont transmis à l'intention du terminal pour sélectionner certains modes de fonctionnement.

BEL	7	Déclenchement d'un signal sonore
BS	8	Déplacement du curseur d'une position à gauche
HT	9	Déplacement du curseur d'une position à droite
LF	10	Déplacement du curseur d'une position vers le bas
VT	11	Déplacement du curseur d'une position vers le haut
FF	12	Effacement de l'écarn
CR	13	Curseur en début de ligne
SO	14	Passage en mode semi-graphique
SI	15	Retour en mode texte
CON	17	Affichage du curseur
REP	18, n	Rèpètition du caractère précédent (n= nombre + 64)
COFF	20	Masquage du curseur
CAN	24	Remplissage par des espaces
SS2	25, a, m	Attribut d'accentuation (a= accent, m=minuscule)
ESC	27, c	Attribut de caractère (c= code)
RS	30	Curseur en colonne 1 de la ligne 1
US	31, x, y	Positionnement du curseur (x et y = coordonnées + 64)
	32	espace (parfois nécessaire à cause du BASIC)

Codes de fonctions gérant les modes d'affichage

Pour être actifs, ceux-ci doivent être précédés du code décimal 27

CODE	FONCTION	CODE	FONCTION
64	caractère noir	79	double taille
65	caractère rouge	80	fond noir
66	caractère vert	81	fond rouge
67	caractère jaune	82	fond vert
68	caractère bleu	83	fond jaune
69	caractère magenta (mauve)	84	fond bleu
70	caractère cyan (bleu ciel)	85	fond magenta
71	caractère blanc	86	fond cyan
72	caractère clignotant	87	fond blanc
73	caractère fixe	89	fin de lignage
76	taille normale	90	début de lignage
77	double hauteur	92	fond normal
78	double largeur	93	inversion de fond

Codes d'accentuation et caractères spéciaux

SEQUENCE	CARACTERE GENERE
25, 65, a	à
25, 65, e	é
25, 65, u	ù
25, 66, e	ë
25, 67, a	â
25, 67, e	ê
25, 67, i	î
25, 67, o	ô
25, 67, u	û
25, 72, e	e
25, 72, i	i
25, 75, c	ç
25,106	Œ
25,122	œ
25, 48	°

Remarque: Un écran VIDEOTEX se compose de 24 lignes de 40 colonnes plus une ligne de service la ligne 0 Les lignes et les colonnes sont respectivement numérotées de 65 à 88 et de 65 à 104 Certaines commandes ne sont prises en compte que si elles sont immédiatement précédées d'un espace Exemple: _ couleur d'un caractère, _ mode souligné



L'interface série est également gérée par le basic

Par rapport à l'EXELBASIC+, l'EXELBASIC-VS apporte quelques instructions supplémentaires à celles décrites dans le manuel EXELBASIC+.

DELETE "10 R" met le fil DSR en sortie à l'état haut (READY)
DELETE "10 B" met le fil DSR en sortie à l'état bas (BUSY)

Différents paramètres de contrôle de l'interface série ont été ajoutés Ce sont des paramètres de lecture des caractères reçus Il est possible de tester la réception d'un ou de deux caractères avant l'attribution du contenu du buffer à la variable basic interressée

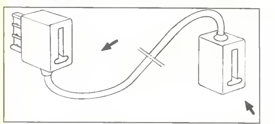
```
OPEN #1,"10 Y=0D, Z=0A",VARIABLE 1000
LINPUT #1,A$
```

Y et Z sont respectivement l'avant-dernier et le dernier caractère avant le retour de la fonction LINPUT La taille du buffer basic de saisie est fixée par l'instruction VARIABLE. 0D et 0A représentent les valeurs hexadécimales respectives des codes RETOUR CHARIOT et LINE FEED Ces valeurs sont données à titre d'exemple. Le code d'erreur err 0,27 est généré si aucun caractère n'a été reçu dans un délai de 30 secondes

Si la réception des caractères correspondent aux paramètres Y et Z conduit à l'affectation de plus de 255 caractères à la variable A$, une erreur basic classique de type "chaine trop longue" sera retourné Il est possible de ne pas utiliser les paramètres Y et Z
Dans ce cas, le contenu du buffer est automatiquement attribué à la variable A$ Si aucun caractère n'est présent dans le buffer , le contenu de A$ sera vide Il est également possible de n'attendre qu'un seul caractère. Dans ce cas, seul le paramètre Z sera à fixer

LIAISON EXELTEL-VS / TERMINAL VIDEOTEX

Ce système est indispensable au programmeur désireux de tester des applications de type serveur Il permet évidemment de s'affranchir totalement des coûts de communication ..! Il suffit d'utiliser les deux prises femelles disponibles La prise mâle reste utilisable et la priorité est donnée au terminal branché sur la prise femelle située à l'autre extrémité de la rallonge Ce type de rallonge coûte environ 80 francs en grande surface.



NOTE: Les données techniques fournies dans le présent banc d'essai sont extraites de la documentation STUM 1B. Vous pouvez vous procurer cette dernière (environ 150 F) en contactant le C N E T Centre PARIS A, 38/40 rue du Général Leclerc 92131 ISSY-LES-MOULINEAUX Tel : (1) 45 29 61 09

INITIATION BASIC



Représentation graphiques des données

Les données traitées dans un programme peuvent être représentées par de petits graphiques. En effet, ces graphiques sont souvent plus faciles à interpréter ou à analyser qu'une interminable suite de nombres. De plus, le fait de représenter graphiquement des valeurs permet d'en saisir en un coup d'oeil la répartition, les écarts, la moyenne. Ces graphiques aussi sont appellés histogrammes ou "camemberts". Un histogramme à bâtons est plus particuliérement approprié pour représenter une à une les différentes données. Un "camembert" est employé lorsque l'on désire obtenir la répartition en pourcentage d'un ensemble de valeurs

Un exemple simple

Entrons dans le vif du sujet. Un commerçant souhaite représenter graphiquement l'état de son stock

Ce programme réduit à sa plus simple expression affiche le nom du produit, sa quantité et enfin une série de caractères représentant le nombre d'articles.

A partir de cet exemple, nous pouvons dégager plusieurs observations

Le format des données

Les données à représenter appartiennent dans le cas présent à un tableau. Dans un cas général, les données à représenter peuvent appartenir à un fichier externe (Exelmémoire,disquettes ou autre) ou bien être directement introduites par l'utilisateur

```
100 '###################################
110 '#          EXEMPLE SIMPLE          #
120 '###################################
130 CLS "Wbb":CALL COLOR("0Wb')
140 DIM TABLEAU$(0,1)
150 DATA TABLE,2,CHAISE,3,TAPIS,6,CANAPE,1
160 DATA FAUTEUIL,8,BUREAU,7
170 FOR I=0 TO 5
180 READ TABLEAU$(1,0),TABLEAU$(I,1)
190 NEXT I
200 LOCATE (5,1)
210 FOR I=0 TO 5
220 PRINT TAB(1);
230 PRINT USING"########",TABLEAU$(I,0);
240 PRINT USING"##",TABLEAU$(I,1)
250 PRINT TAB(15),RPT$("*",VAL(TABLEAU$(I,1)))
260 NEXT
```

La validité des données

Ne peuvent être représentées que les données numériques. Dans le cas présent, les données sont bien des données numériques. Dans un autre cas, il faut prendre garde à ce que les données soient exploitables. Allez donc représenter graphiquement le mot "ordinateur"!!!

Le choix du mode graphique

Deux modes graphiques sont généralement disponibles sur un ordinateur. L'Exeltel posséde le mode texte et le mode haute résolution. Pour dessiner un graphique à bâtons, il est préférable d'utiliser le mode texte. En effet, ce genre de graphique peut se représenter par une suite de caractéres (carrés pleins ou étoiles par exemple) Pour tracer un camembert, il est préférable d'utiliser le mode haute résolution (mapping)

La représentation proprement dite.

Elle est trés facile à mettre en oeuvre car il suffit d'afficher autant de caractères que la valeur de la donnée. Ainsi, pour une valeur qui vaut 12, nous obtiendrons 12 caractères. De la même manière, en haute résolution, une valeur de 12 sera représentée par 12 points soit 12 pixels

Cet exemple est schématisé à l'extrême car nous avons choisi des valeurs qui peuvent être représentées dans la page texte. Il en va tout autrement dans la réalité

La représentation graphique des données peut se réaliser dans le sens vertical ou horizontal. Dans notre exemple, nous avons choisi le sens horizontal afin d'écrire la référence de l'article et sa quantité



Application à des problèmes généraux

Dans les cas généraux et usuels, les données sont loin d'être aussi faciles à représenter. En effet, il peut exister au sein d'une même série de valeurs des écarts extrêmement importants. Prenons par exemple le cas des différents PNB. Sur les quelques 200 nations, certaines ont des PNB très importants (USA, URSS, etc...), d'autres au contraire ont des PNB très faibles (république d'Haïti, etc) Il devient alors très difficile de représenter les deux extrêmes sur un même graphique

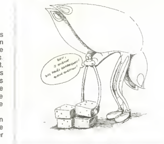
L'ordre de grandeur des données est aussi un frein important à la réalisation de ce genre de graphique. En effet, si nous devons représenter des quantités de l'ordre du million, du milliard ou plus, de sérieux problèmes quant au choix d'une échelle idoine se poseront

Une fois ce problème réglé, il reste encore à traiter la finesse de représentation. Ainsi, si l'on utilise la page texte pour représenter des données, il faudra prévoir une analyse afin de pouvoir représenter des quantités très faibles. Une approximation au dixième de caractère (c'est-à-dire de l'ordre du pixel) doit être sérieusement envisagée.

Le nombre des valeurs à représenter est aussi un handicap. En effet,il est impossible d'afficher deux cents valeurs sur une même page graphique (en mode texte). Il faudra donc prévoir de segmenter la liste de valeurs en autant de pages que nécessaire. Si la représentation s'effectue en mode haute résolution, 100 à 200 valeurs peuvent être représentées. Toutefois, la surface graphique utilisée pour représenter une valeur est faible (largeur de l'ordre du pixel) . De plus, il est de règle de représenter graphiquement des valeurs à l'aide de pavés graphiques assez larges. Même si les problèmes semblent importants au départ, nous allons peu à peu analyser ces derniers et apporter une solution correcte

Le nombre de données

Problème épineux en vérité. En effet, on ne sait pas a priori combien de données devront être représentées, de quelques unes à quelques centaines. Il nous faut donc établir un programme





qui sache organiser en différentes "pages" les données à représenter Même dans le cas d'une représentation graphique en mode haute résolution, le problème du nombre de valeurs peut se poser

D'où proviennent les données ?

Nous parlons bien de valeurs, mais au juste, d'où proviennent les valeurs. ? Sont-elles internes à un programme (cas d'un tableau écrit sous forme de data) ? Sont-elles introduites au cours du déroulement d'un programme (instruction INPUT ou LINPUT puis stockées dans un tableau) ? Sont-elles externes à un programme (cas d'un fichier) ?. En fait, peu importe la provenance des données, car nous traiterons ces dernières comme issues d'un tableau. Oui, en effet, même en considérant les données comme provenant d'une source externe ou d'une introduction manuelle, il faudra organiser ces données sous une format exploitable facilement Le format le plus facile à exploiter reste le tableau.

Les données sont-elles exploitables ?

Pour représenter des données, encore faut-il qu'elles soient représentables. Les données doivent être des repréesentetions numériques valides (autrement dit des nombres).

L'organisation des données en tableaux.

Si remplir un tableau avec des données ne pose pas de problèmes, organiser le tableau en plusieurs "pages" devient plus pointu. En effet, en plus de la segmentation en "pages" proprement dit, il faut encore gérer la sélection de ces différentes pages.

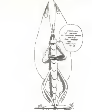
Techniques de bases:

Pour segmenter un tableau en plusieurs pages, il nous faut connaître le nombre d'éléments que comportera chaque page. Selon le nombre de pages qui peut varier en fonction de la dimension propre du tableau, on utilise différents pointeurs qui nous fourniront en permanence les indications sur la page courante.

Paramètres de départ

N= Nombre d'éléments du tableau
X= Nombre d'éléments de chaque page
INDICE = Nombre de pages du tableau

A partir de ces éléments, nous pouvons déterminer le nombre de pages que comportera le tableau. Ce nombre de pages est donné par :



INDMAX=INT((N-1) / X)

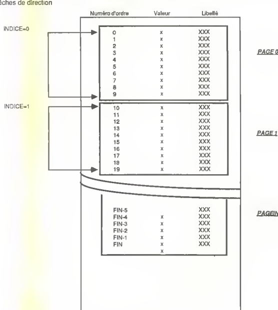
Ainsi, pour un tableau comportant 62 éléments et un nombre d'éléments par page de l'ordre de 10 noue obtenons **INDMAX**=INT(62/10)=6 . Les données se répartiront en 7 pages, de la page 0 à la page 6. Donc **INDMAX** nous donne le numéro de la dernière page qui ne comporte que 2 éléments. Le nombre d'éléments restant dans la dernière page est donné par **N-(INDMAX*10)**.

Maintenant que nous avons défini ces numéros de pages, il nous reste à établir la procédure de segmentation. Quel est le problème ? Nous voulons sélectionner et traiter les données d'une page d'un tableau. Pour sélectionner une page, nous avons besoin d'un programme qui "tourne" les pages comme dans un livre ou un cahier. Nous connaissons le nombre de pages, le nombre d'éléments par page et même les numéros de pages Forts de ces précieux renseignements,

nous pouvons proposer la solution suivante:

Une "fenêtre" de lecture des données peut se déplacer sur l'étendue du tableau. Cette "fenêtre" autorise la lecture de 10 éléments. Pour déplacer cette "fenêtre" de lecture, nous utiliserons les flèches de direction

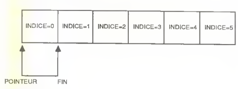
Pour réaliser la lecture des éléments, nous devons à tout moment connaître le numéro de la page courante Toute la difficulté consiste à gérer au mieux cette lecture tout en interdisant l'accès après la dernière page et avant la première page.



INITIATION BASIC

Valeur de début de boucle et de fin de boucle

Pour lire les données d'une page, nous utilisons une boucle dont l'indice variera du premier élément de la page courante jusqu'au dernier élément de la page courante.
La première page du tableau sera la page numéro 0, la dernière page du tableau est **INDMAX.** Ces valeurs sont les valeurs extrêmes. Il ne reste plus qu'à définir les valeurs de début et de fin de boucle pour chaque page. Ces valeurs dépendent du nombre d'éléments par page.
Si X représente le nombre d'éléments par page, la valeur de début de boucle est **INDICE*10**. En effet, si indice vaut 0, la valeur de début de boucle vaut toujours 0.
Si nous prenons **INDICE=INDMAX** (cas d'un tableau de 62 éléments), la valeur de début de boucle est 60. Cette valeur de début de boucle sera contenue dans la variable **POINTEUR**. Une fois la valeur de début de boucle connue, il nous reste à déterminer la valeur de fin de boucle. Cette valeur de fin de boucle est égale à la valeur de début de boucle augmentée de la valeur représentant le nombre d'éléments par page. Dans le cas qui nous occupe, cette valeur vaut X. Il reste un cas particulier à traiter. En effet, la dernière page de notre tableau peut contenir



POINTEUR=INDICE *X
FIN=POINTEUR+X-1

Remplaçons X par sa valeur et nous obtenons les différentes pages segmentées

SI INDICE =0 ALORS POINTEUR = 0 et FIN = 9
SI INDICE =1 ALORS POINTEUR = 10 et FIN= 19
SI INDICE =2 ALORS POINTEUR = 20 et FIN = 29
etc

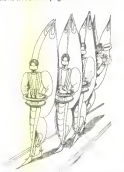
Si le nombre d'éléments total du tableau est 62 par exemple, le dernier numéro de page est 6, donc INDICE=INDMAX=6
SI INDICE = 6 ALORS POINTEUR = 60 et FIN = 62
En effet, nous avons postulé que lorsque nous sommes en présence de la dernièer page, la variable FIN prend comme valeur le nombre total d'éléments, ici 62.

*IMPORTANT: Le BASIC majore de 1 la valeur de fin de boucle. Exemple:
FOR I = 0 TO 10: NEXT
Si l'on demande PRINT I à la sortie de la boucle, on obtient I=11. Il faut prendre soin de retrancher 1, pour obtenir la valeur réelle de l'indice.

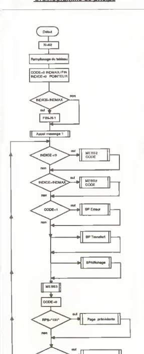
moins de X éléments. Dans cette situation, une lecture de données générera une erreur. Nous sommes donc obligés d'attribuer une autre valeur de fin de boucle lorsque la lecture des données s'effectue pour la dernière page du tableau. Cette valeur de fin de boucle pour la dernière page est facile à déterminer. En effet elle est donnée par le nombre d'éléments total du tableau soit N. La variable **FIN** sera chargée de contenir cette valeur de fin de boucle.

Le défilement des pages

Nous avons déterminé les valeurs de début de boucle **POINTEUR** et les valeurs de fin de boucle **FIN**. Ces deux limites permettront d'effectuer la lecture des données ou plutôt l'affichage des données. Maintenant que nous connaissons ces deux valeurs, nous devons gérer la sélection des pages du tableau. Deux solutions peuvent être envisagées: sélectionner une page en tapant son indice ou sélectionner une page en "rotatif", c'est-à-dire en faisant défiler les pages. C'est la deuxième solution que nous retiendrons car elle correspond beaucoup plus à la réalité des choses. En utilisant deux touches, une pour le défilement avant, l'autre pour le défilement arrière, nous afficherons à l'écran la page désirée.



Ordinogramme de pricipe



Pour réaliser ce défilement, nous connaissons en permanence le numéro de la page donnée par la variable **INDICE.** Cette variable sera incrémentée ou décrémentée en fonction du mode de défilement choisi (avant ou arrière). Néanmoins, nous devons avoir à l'esprit que nous devons interdire le défilement avant lorsque nous pointons sur la première page ainsi que le défilement arrière lorsque nous pointons sur la dernière page. De plus, il faut que l'affichage des différentes pages commence toujours avec la première page, c'est-à-dire la page 0 (INDICE=0). Pour réaliser cette dernière condition, il suffit de préciser au départ les paramètres **INDICE=0, POINTEUR=INDICE*10 et FIN = POINTEUR+9.**

L'interdiction de défilement

Cette section est légèrement plus difficile à mettre en place. De quoi avons-nous besoin pour interdire le défilement avant la première page et après la dernière page ?
En premier lieu, nous devons tester les différentes limites qui provoqueront l'interdiction. Un simple test peut suffire:
IF INDICE < 0 THEN . .....
IF INDICE>INDMAX THEN .....
En fonction du résultat du test, le programme se branche à un sous-programme responsable d'afficher un message d'erreur. Ce même sous-programme gère un indicateur d'état. Cet indicateur (CODE) est positionné à 1 lorsque le programme passe par ce sous-programme. Cet indicateur d'état est ensuite utilisé dans le déroulement du programme pour mettre à jour les différents paramètres (INDICE , POINTEUR et FIN) en fonction du choix de défilement proposé. En effet, si l'indicateur d'état passe à 1, le programme se branche à un sous-programme qui restitua la valeur initiale (la valeur avant l'occurance de l'erreur) pour INDICE, POINTEUR et FIN.

Traitement de la page précédente

Ce sous-programme est chargé de gérer le défilement à "rebours" si l'on peut dire. Ce sous-programme permet de mettre à jour les paramètres comme INDICE, POINTEUR et FIN. Le paramètre INDICE est décrémenté d'une unité et un indicateur d'état DRAPEAU est positionné. Cet indicateur renseigne le programme sur le type de défilement effectué (avant ou arrière). Cet indicateur est ensuite utilisé pour gérer les interdictions de défilement (voir ci-avant)

Traitement de la page suivante

Ce sous-programme est chargé de gérer le défilement en avant . Ce sous-programme permet de mettre à jour les paramètres comme INDICE, POINTEUR et FIN. Le paramètre INDICE est décrémenté d'une unité et un indicateur d'état DRAPEAU est positionné. Cet indicateur renseigne le programme sur le type de défilement affectué (avant ou arrière) . Cet indicateur est ensuite utilisé pour gérer les interdictions de défilement (voir ci-avant).

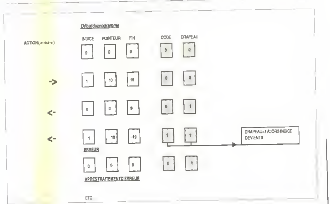
Débordement de page

En fonction de la valeur de DRAPEAU, INDICE, POINTEUR et FIN sont mis à jour.

Transfert dans une page

Ce sous-programme exploite directement les paramètres POINTEUR et FIN pour transférer les différentes valeurs dans une page. Cette page est en fait une réplique du tabeau original. POINTEUR et FIN définissent les limites de la page





Ordinogramme de principe

L'ordinogramme de principe ne pose aucune difficulté. Tel qu'il est construit, le programme pourra supporter l'appel d'autres sous-programmes. La structure modulaire des programmes prend ici toute sa valeur. En effet, maintenant que nous savons segmenter en différentes pages un tableau principal, nous pouvons aborder la partie représentation graphique.

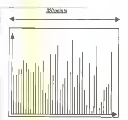
Ecran texte ou écran graphique?

L'éternel problème. Doit-on représenter les données en utilisant l'écran semi-graphique encore appelé texte ou l'écran haute résolution appelé écran graphique? Tout dépend de ce que vous attendez d'un histogramme. L'écran graphique permet de représenter facilement sur une même page trois cents valeurs environ. Le problème du choix d'une échelle n'est pas un gros problème. En effet, des valeurs avec des écarts de l'ordre d'un facteur 200 peuvent être représentées sans avoir à "bricoler".

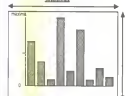
Ce n'est malheureusement pas le cas avec la page semi-graphique. Mais, il y a un mais, la surface de représentation d'une valeur est réduite au pixel, ce qui, avouons-le, n'est guère visible. Si nous utilisons la page semi-graphique, la surface de représentation d'une valeur est de l'ordre de 8 pixels, ce qui est tout de même plus visible.

Optimalisation de l'écran texte ou mapping

Concept de taille écran utile.
L'écran texte ou l'écran haute résolution offrent des surfaces de travail différentes.
Pour déterminer la surface de travail, nous devons tenir compte de différents paramètres comme l'espace entre deux valeurs à représenter. Même en utilisant l'écran haute résolution, on ne pourra représenter plus de 150 valeurs à l'écran. En effet, en laissant un pixel entre deux valeurs et en prenant un pixel comme unité de représentation graphique, nous obtenons 300 pixels. Ensuite, nous devons tenir compte du tracé de l'axe vertical et de l'axe horizontal, ainsi que de la graduation. Il



L'écran graphique permet de représenter plus de valeurs que l'écran semi-graphique. Les graduations ne peuvent pas être représentées



L'écran semi-graphique est limité quant au nombre de valeurs représentées. La graduation peut être par contre représentée

est à noter que si nous ne possédons pas de programmes utilitaires pour écrire du texte dans la page haute résolution, nous ne pourrons pas représenter de graduation, ce qui assez ennuyeux

L'écran semi-graphique peut être un bon compromis car il offre la possibilité d'écriture de texte et , moyennant quelques astuces nous pouvons représenter des valeurs avec des écarts d'un facteur 170 à 180, pratiquement de l'ordre du pixel Evidemment, nous ne pouvons pas représenter plus de 10 valeurs par page d'affichage Tout comme pour un écran graphique, nous devons laisser un caractère d'espace entre deux valeurs, réserver l'emplacement pour tracer l'axe horizontal et l'axe vertical, et inscrire la graduation.

Ne vous imaginez surtout pas que nous avons surmonté les difficultés. Bien au contraire, nous allons aborder les problèmes pointus.

Les différentes échelles de représentation

Dans tous les problèmes de représentation graphique, intervient tôt ou tard le choix d'une échelle correcte En effet, il est peu probable que les valeurs qui doivent être exploitées puissent parfaitement s'inscrire dans l'écran graphique ou semi-graphique

Comment définir l'échelle ? Dans la pratique, nous devons tenir compte de deux facteurs primordiaux. le maxima et le minima des valeurs Mais ce n'est pas tout. Les valeurs traitées sont-elles positives, négatives, voisines de zéro ou au contraire infiniment grandes ? Joli casse-tête en perspective



Pour ce qui concerne le minima et le maxima d'une série de valeurs, le cas a été traité dans Exelement Vôtre numéro 14, nous ne reviendrons pas sur ce problème Mais avant de continuer, analysons le problème plus en profondeur

Nous devons nous poser plusieurs questions

-Les données sont-elles toutes positives ?

-Les données sont-elles toutes négatives ?

-Les données sont-elles à la fois positives et négatives ?
Si les données sont toutes positives ou toutes négatives, les valeurs pourront être représentées dans le même sens Si, au contraire les données sont positives et négatives, les valeurs doivent être représentées de part et d'autre d'un axe médian Dans cette hypothèse, l'écart entre les minima et les maxima doit être réexaminé.

Recherche du signe des valeurs

Tout comme la recherche du minima et du maxima, nous pouvons connaître le signe de chaque donnée La fonction SGN de l'Exelbasic retourne la valeur -1 si l'expression numérique fournie est négative et 1 si l'expression numérique fournie est positive
Le programme chargé de déterminer le signe des valeurs peut être inclus dans le programme de recherche des maximas et des minimas Comme nous l'avons écrit plus haut, trois cas peuvent se présenter

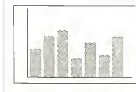
-Les valeurs sont toutes positives Dans ce cas précis, un indicateur d'état (que nous appellerons ETAT) est forcé à 1 En fonction de cet indicateur, nous utiliserons pour la suite du programme un système d'axe, une graduation et une couleur de représentation fixés par avance

-Les valeurs sont toutes négatives Dans ce cas précis, un indicateur d'état est forcé à 2 Nous utiliserons le même système d'axe que précédemment Seule la graduation et la couleur de représentation changeront

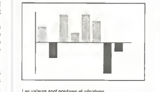
-Les valeurs sont négatives ou positives L'indicateur est forcé à 3 Dans ce cas, nous utiliserons un axe médian sur lequel les valeurs seront réparties de part et d'autre Deux couleurs de représentation seront utilisées

Pour réaliser cette partie de programme, nous procéderons par élimination successive des différents cas.

1-Les valeurs sont-elles toutes positives ?
Dans une boucle variant de 0 à N-1, le signe de chaque valeur est comparé avec -1. Si cette condition est remplie, cela signifie qu'au moins une valeur de la liste est négative. Si la condition



Les valeurs sont totalement positives ou totalement négatives



Les valeurs sont positives et négatives

n'est pas remplie, alors toutes les valeurs sont positives et l'indicateur ETAT est forcé à 1
Si la condition est remplie, cela ne signifie pas pour autant que toutes les valeurs soient négatives.

2-Les valeurs sont-elles toutes négatives ?

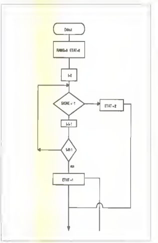
On répète la même opération que décrite ci-dessus mais l'on change la condition Le signe de chaque valeur est comparé avec 1 Si la condition est remplie, cela signifie qu'au moins une valeur est positive Si la condition n'est pas remplie, alors toutes les valeurs sont négatives et l'indicateur ETAT est forcé à 2
Si la condition énoncée ci-dessus est vraie, alors les valeurs sont positives ou négatives L'indicateur ETAT est forcé à 3.

A la sortie de ce bout de programme, l'indicateur ETAT peut prendre quatre valeurs 0,1,2,3

INITIATION BASIC

0 -> Toutes les valeurs sont nulles
1 -> Toutes les valeurs sont positives
2 -> Toutes les valeurs sont négatives
3 -> Les valeurs sont positives ou négatives

Maintenant, en fonction de l'indicateur ETAT, le programme peut se brancher à différents sous-programmes. Ces sous-programmes permettent de déterminer les minimas et les maximas pour les différents cas envisagés. Une instruction de branchement multiple suffit (ON ETAT GOSUB).



Recherche des minimas et maximas

Si les valeurs sont toutes positives ou toutes négatives, la recherche du minima et du maxima sera identique dans l'absolu. Toutefois, le minima positif devient le maxima négatif et le maxima positif devient le minima négatif.



Par contre, si les valeurs sont panachées, le problème devient un peu plus complexe à résoudre. En effet, les valeurs seront représentées de part et d'autre d'un axe médian. Cela implique une recherche de deux maximas et deux minimas: Le minima et le maxima positif et le minima et le maxima négatif. La recherche du minima et du maxima positif doit exclure toutes les valeurs négatives. De même, la recherche du minima et du maxima négatif doit exclure toutes les valeurs positives.

A la sortie de ce bout de programme, nous avons tous les éléments nécessaires pour définir les différentes échelles de représentation.

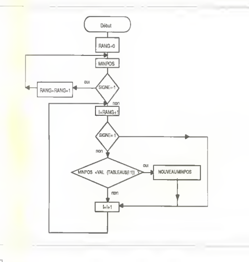
Les échelles

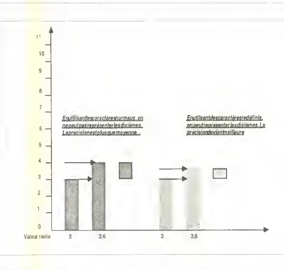
En fonction des minimas et des maximas obtenus, nous pouvons déterminer les rapports entre ces quantités (n'oublions pas qu'il existe des minimas et des maximas positifs et négatifs). Ces rapports sont par la suite testés afin de vérifier si les valeurs d'une série peuvent être représentées à l'écran sans subir d'approximation globale.
Si les valeurs sont toutes positives ou toutes négatives, le RAPPORT s'obtiendra en effectuant: **RAPPORT=MAX / MIN**. Ce rapport est ensuite comparé avec la taille utile de l'écran. Dans le cas d'un écran semi-graphique, en tenant compte de la représentation des différents axes et messages, la taille a été ramenée à 170 pixels. Si

INITIATION BASIC

le **RAPPORT** est supérieur à cette limite, les valeurs ne pourront pas être représentées à l'écran sans subir de procédures d'approximation.
L'échelle de représentation s'obtient en divisant (pour un écran texte) le nombre de caractères qui peuvent être représentés par le maxima obtenu.
Ainsi, pour l'écran semigraphique nous obtenons: **ECH= 17 / MAX.** Cette échelle constitue un facteur multiplicateur qui sera appliqué aux valeurs réelles.
Pour ce qui concerne les valeurs panachées (négatives et positives), il faudra déterminer deux rapports **RAPPORTPOS= MAXPOS/MINPOS** et **RAPPORTNEV=MAXNEV/MINNEV.** Ces deux rapports sont ensuite comparés avec la taille de l'écran utile qui, dans ce cas, est réduite de moitié. Si l'un des deux rapports est supérieur aux 80 pixels alloués pour la représentation, les valeurs ne pourront pas être représentées sans subir de procédure d'approximation.
Si les deux rapports satisfont à la condition, les deux rapports sont comparés entre eux. Le plus grand des deux est choisi comme référence et l'échelle de représentation est donnée par : **ECH= 8/ MAXPOS <u>ou</u> MAXNEV.**
Pour réaliser cette partie du programme qui sera, comme vous vous en doutez, un ensemble de sous-programmes, nous nous servirons de l'indicateur ETAT. En effet, en fonction de cet indicateur, nous connaissons la répartition des différentes valeurs selon leurs signes.

Si les valeurs ne peuvent être représentées, ces dernières doivent subir des procédures d'approximation. Cette situation peut se produire lorsque les écarts entre les différentes valeurs sont extrémement importants (facteur 180).
Nous ne traiterons pas ce cas dans notre analyse





L'affichage et les différents éléments de l'affichage

L'affichage doit comprendre un système d'axe, la graduation de l'axe vertical, la référence de la page en cours et les différents bâtons représentant les valeurs

Définition des différents caractères

Le travail dans le mode semi-graphique impose une redéfinition de certains caractères Ces caractères seront utilisés pour représenter l'axe vertical et permettront de travailler au pixel près
En effet, le mode semi-graphique impose un travail par bloc de 8 x 10 points Il est donc impossible en utilisant le jeu de caractères standard de pouvoir représenter une différence d'un pixel Par contre, en définissant une vingtaine de caractères, il est possible de simuler cette différence
Pour tracer les axes, nous utiliserons trois caractères qui seront définis par.

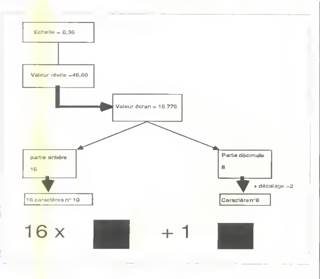
```
CALL CHAR (20,"FF0000000000000000FF")
CALL CHAR(21,"FF000000000000000000")
CALL CHAR(22,"80808080808080808080")
```

Les autres caractères qui seront définis à partir du numéro 1 seront utilisés pour représenter les caractères au dixième
Nous avons dit précédemment que les valeurs sont représentées sur 170 pixels dans le cas où ces dernières possèdent le même signe En redéfinissant 10 caractères, il est possible de représenter les valeurs positives ou négatives sur 170 pixels C'est ce que nous explique la figure ci-contre

Comment représenter des écarts d'un pixel?

Soit les valeurs réelles à représenter 46,60 et 17,35. Le rapport vaut 12,69 et l'échelle vaut 0,36. Le rapport nous indique que les valeurs peuvent être représentées sans approximation. Les valeurs qui doivent être tracées à l'écran valent 46,60 x 0,36 =16,77 et 12,69 x 0,36 = 4,57.
Nous partons du principe que, pour représenter les données, nous attichons autant de caractères que de valeurs. Dans notre cas 16,77 et 4,57. Mais le basic ne sait pas représenter en mode texte 16,77 ou 4,57. Il interprétera 16,77 comme 16 et 4,57 comme 4 et attichera 16 et 4 caractères.

Toutefois, il est possible de contourner cette difficulté. Nous savons que la taille utile de l'écran texte est de 170 pixels. Un caractère vaut 10 pixels en hauteur. 17 caractères peuvent donc être représentés. Entre le 16 ème caractère et le 17 ème caractère, il existe une différence de 10 pixels. Cette différence peut être exploitée pour représenter les valeurs 16,1-16,2-16,3-16,4-16,5-16,6-16,7-16,8-16,9. Dans le cas qui nous occupe, la valeur écran vaut 16,77 en fait 16,8 si cette valeur est arrondie au premier chiffre significatif après la virgule. Nous pouvons donc imaginer un caractère qui comporte 8 pixels. Ce caractère sera juxtaposé aux 16 premiers caractères. De cette manière, nous aurons une représentation assez fidèle de la valeur 16,77



Principe de fonctionnement de la représentation au pixel

Pour chaque valeur du tableau, nous calculons dans un premier temps sa valeur absolue
HIST=ABS(VAL(PAGE$(I,1)))
Nous calculons ensuite la partie entière de la valeur écran.
ECRAN=INT(HIST*ECH)
Nous calculons ensuite la partie décimale
RESTE=INT(10*(HIST*ECH-ECRAN))
A ce stade, nous obtenons deux valeurs distinctes **ECRAN** et **RESTE**. La première valeur peut varier entre 0 et 17 et donne le nombre de caractères de 10 pixels de haut. La deuxième valeur, **RESTE**, peut varier entre 0 et 9 et donne le caractère au dixième qu'il faut juxtaposer aux caractères de 10 pixels.
Le problème qui se pose maintenant consiste à sélectionner directement le caractère correspondant à la valeur **RESTE** tout en sachant que nous devons utiliser deux séries de caractères "au dixième". Une série pour les valeurs positives ou négatives (indicateur ETAT = 1 ou 2) et une série pour les valeurs négatives (indicateur ETAT= 3). Les différents caractères redéfinis sont numérotés de 1 à 9 pour les caractères au dixième côté positif/négatif (indicateur ETAT= 1 ou 2) et de 11 à 19 pour les caractères au dixième côté négatif (indicateur ETAT = 3)
Le premier jeu de caractères peut être sélectionné directement en utilisant la variable **RESTE** pour indice. Par exemple, si nous avons RESTE = 8, PRINT CHR$(RESTE) permet d'afficher le caractère correspondant à 8/10 ème. Par contre, pour sélectionner le deuxième jeu de caractères, il faut introduire un décalage. Ce décalage est égal à 10. Ainsi, pour rendre compatibles les deux situations, nous définissons une variable appelée GRAPHIC qui dans le premier cas donne **GRAPHIC=RESTE,** et dans le deuxième cas **GRAPHIC = RESTE + 10**. Les caractères correspondants à la partie décimale seront affichés en effectuant **PRINT CHR$(GRAPHIC).**
Lorsque RESTE vaut 0 ou ECRAN vaut 0 , une situation bien particulière est générée. Cette situation est traitée en détail un peu plus loin dans ce chapitre.

Nous pouvons écrire un sous-programme de redéfinition des caractères qui sera appelé au départ du programme.


Caractère 1/10

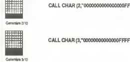
CALL CHAR (1,"000000000000000000FF")


Caractère 2/10

CALL CHAR (2,"0000000000000000FFFF")


Caractère 3/10

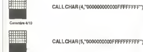
CALL CHAR (3,"00000000000000FFFFFF")


Caractère 4/10

CALLCHAR (4,"000000000000FFFFFFFF")

Caractère 5/10

CALLCHAR(5,"0000000000FFFFFFFFFF")


Caractère 6/10

CALLCHAR(6,"00000000FFFFFFFFFFFF")


Caractère 7/10

CALLCHAR(7,"000000FFFFFFFFFFFFFF")

Caractère 8/10

CALLCHAR(8,"0000FFFFFFFFFFFFFFFF")


Caractère 9/10

CALLCHAR(9,"00FFFFFFFFFFFFFFFFFF")


Caractère 10/10

CALLCHAR(10,"FFFFFFFFFFFFFFFFFFFF")



```
CALL CHAR (11,"FF00000000000000")
```
Caractère 1/10

```
CALL CHAR (12,"FFFF000000000000")
```
Caractère 2/10

```
CALL CHAR (13,"FFFFFF0000000000")
```
Caractère 3/10

```
CALL CHAR(14,"FFFFFFFF00000000")
```
Caractère 4/10

```
CALLCHAR(15,"FFFFFFFFFF000000")
```
Caractère 5/10

```
CALLCHAR(16,"FFFFFFFFFFFF0000")
```
Caractère 6/10

```
CALLCHAR(17,"FFFFFFFFFFFFFF00")
```
Caractère 7/10

```
CALLCHAR(18,"FFFFFFFFFFFFFFFF0000")
```
Caractère 8/10

```
CALLCHAR(19,"FFFFFFFFFFFFFFFFFF00")
```
Caractère 9/10

Le tracé des axes

Ensuite, nous pouvons écrire le sous-programme chargé de tracer les axes. Ici encore, nous devons tenir compte de la valeur de l'indicateur ETAT. En effet, nous aurons deux représentations d'axes possibles. Si ETAT vaut 1 ou 2, l'axe vertical n'est pas partitionné, si ETAT vaut 3, l'axe vertical devra tenir compte de la partie positive et la partie négative.

Les graduations

Pour simplifier le problème, nous indiquerons le maxima arrondi à la dizaine, à la centaine ou au millier supérieur , le zéro et une graduation médiane. Comme nous ne savons pas a priori l'ordre de grandeur du maxima (mille, dix mille ou plus), nous n'afficherons que deux chiffres. Un facteur de multiplication sera inscrit au haut de la page. Comme pour les autres cas, en fonction de l'indicateur ETAT, plusieurs sous-programmes doivent être mis en place.

Calcul du maxima arrondi pour les valeurs positives ou négatives

En fonction des valeurs à représenter nous devons tenir compte de plusieurs facteurs.

1) Les valeurs sont positives et supérieures à 10
2) Les valeurs sont négatives et inférieures à -10
3) Les valeurs sont positives et comprises entre 0 et 1
4) Les valeurs sont négatives et comprises entre -1 et 0

Les deux premiers cas pourront être traités de la même manière. Les cas 3 et 4 pourront eux aussi être traités de la même manière mais d'une façon tout à fait disjointe de la première.

Cas 1 et 2

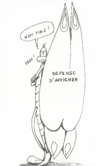
Toute la difficulté consiste à obtenir l'arrondi le plus proche pour toute valeur. Nous devons avoir à l'esprit que nous désirons donner un ordre de grandeur et non une grandeur précise.
Pour réaliser l'approximation, nous considérerons le nombre comme une chaîne de carcatères. En effet, chaque terme d'une chaîne de caractères

INITIATION BASIC

peut être segmenté. Comme l'indique la figure ci-après, pour tous les nombres supérieurs à 10, l'arrondi peut être calculé sur le deuxième chiffre. Si les nombres sont supérieurs à 10, seule la partie entière est conservée.
Ce deuxième chiffre peut être isolé comme un caractère, puis traité comme un chiffre.
Le maxima d'une série de valeurs est d'abord converti en chaîne de caractères.
MAX$=STR$(INT(ABS(MAX)))
Le chiffre est isolé
RANG$=SEG$(MAX$,DICHO,1):
RANG=VAL(RANG$)



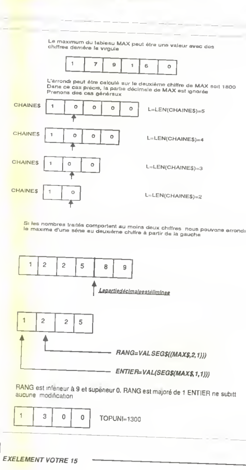
Chaque nombre traité peut être considéré comme la somme de plusieurs chiffres élevés à différentes puissances de 10. 1275 peut s'écrire comme $1 \times 10^3 + 2 \times 10^2 + 7 \times 10^1 + 5 \times 10^0$.
Le maxima d'une série peut donc être décomposé de cette manière et nous pouvons connaître le facteur maximal de multiplication. Si L est la longueur maximale de la chaîne de caractères représentant le maxima, le facteur maximal vaut 10^{L-1}. Si par exemple MAX=1275, L vaut 4, FACTEUR vaut $10^{(4-1)}$ soit 10^3. Pour cette valeur de MAX, un arrondi à 1300 est acceptable.
Si nous arrondissons à partir du deuxième chiffre (RANG), nous devons aussi connaître le premier chiffre (ENTIER) qui peut éventuellement changer. Si MAX vaut 1950 , l'arrondi sera à 2000.
ENTIER=SEG$(MAX$,1,1) .
MAX peut être exprimé dans un premier temps par la suite de facteurs:
MAX # ENTIER*10^{MULT}+RANG*10^{MULT-1}
En fonction de **RANG,** plusieurs actions peuvent être entreprises. Si RANG vaut 0, la valeur de MAX est choisie comme valeur arrondie et la valeur maximale de la graduation TOPUNI vaut ENTIER*10^{MULT} + RANG*10^{MULT-1}.
Si RANG est supérieur à 0 mais inférieur à 9, seul RANG est majoré de 1 et TOPUNI devient ENTIER*10^{MULT}+ (RANG+1) *10^{MULT-1}.
Si rang vaut 9, alors ENTIER est majoré de 1 tandis que RANG prend la valeur 1. TOPUNI devient (ENTIER + 1)*10^{MULT}+1*10^{MULT-1}
Le système a ses limites: En effet, si MAX vaut 162, la valeur arrondie deviendra 170. Plus les nombres sont grands ou petits, plus la précision de l'arrondi est effective.



Cas 3 et 4

Les cas 3 et 4 peuvent être traités de la même manière si un petit traitement leur est appliqué.

En fait, si MAX est inférieur à 10, il suffit de le multiplier autant de fois que nécessaire pour obtenir une valeur supérieure à 10. Un compteur Z enregistre le nombre de multiplications effectué.


Le maximum du tableau MAX peut être une valeur avec des chiffres derrière la virgule
1
7
9
1
6
0
L'arrondi peut être calculé sur le deuxième chiffre de MAX soit 1800
Dans ce cas précis, la partie décimale de MAX est ignorée
Prenons des cas généraux
CHAINE$
1
0
0
0
0
L=LEN(CHAINE$)=5
CHAINE$
1
0
0
0
L=LEN(CHAINE$)=4
CHAINE$
1
0
0
L=LEN(CHAINE$)=3
CHAINE$
1
0
L=LEN(CHAINE$)=2
Si les nombres traités comportent au moins deux chiffres nous pouvons arrondir le maxima d'une série au deuxième chiffre à partir de la gauche
1
2
2
5
8
9
Lapartiedécimaleestéliminee
1
2
2
5
RANG=VALSEG$((MAX$,2,1)))
ENTIER=VAL(SEG$(MAX$,1,1)))
RANG est inférieur à 9 et supérieur 0. RANG est majoré de 1 ENTIER ne subitt aucune modification
1
3
0
0
TOPUNI=1300

Ce compteur permettra de déterminer le facteur de multiplication. Le reste de l'opération d'arrondi reste strictement identique à celui décrit plus haut. Si par exemple MAX vaut 0,0198, MAX doit être multiplié par 1000 pour satisfaire aux conditions d'arrondi. Le compteur de multiplication par 10 vaudra 4 après cette opération. Le facteur de multiplication sera donc de 0,001. En effet, si nous avons multiplié MAX par 1000 pour obtenir des conditions de traitement, nous devons signifier à l'utilisateur que l'unité de graduation doit être multipliée par un facteur 10^{-3}

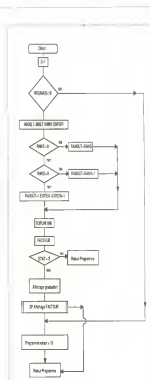
Le cas des valeurs positives et négatives peut être traité par ce programme. Seul l'affichage de la graduation est différent.

L'affichage de la graduation

Nous avons dit que nous afficherons trois graduations: le zéro, une graduation médiane et la graduation maximale. Comme la place réservée aux graduations n'est pas importante, nous n'afficherons au plus que 3 chiffres. Un facteur de multiplication sera affiché en haut de la page.

L'affichage des differentes pages

Toute la difficulté consiste à bien poser le problème. Nous connaissons le nombre de pages à afficher. Ce nombre de pages est donné par INDMAX. Le numéro de chaque page est donné par INDICE. Nous savons que la valeur POINTEUR peut varier de 1 à un nombre appelé FIN identifiant la fin d'une page. Si nous voulons utiliser la même boucle pour lire les données et afficher la représentation des données, nous devons introduire une nouvelle notion de décalage. Il faut trouver une relation entre la valeur de POINTEUR et la position d'affichage. Dans le cas présent, le nombre d'éléments par page est fixé à 10. Cela signifie que toutes les pages traitées ne pourront comporter que 10 éléments. Autrement dit, Si pointeur vaut 15, cela signifie que nous sommes en présence du 6ème élément de la page 1 (la première page est assimilée à la page 0). En effet, la page 0 contient les éléments 0, 1, 3, 4, 5, 6, 7, 8, 9, la page 1 contient les éléments 10, 11, 12, 13, 14, 15, etc. Dans la représentation graphique des données, l'élément 5 sera représenté à la même position que l'élément 15 ou l'élément 25 ou l'élément 35. La relation entre les éléments et



leur position d'affichage est facile, c'est le produit entre le numéro de page (INDICE) et le nombre d'éléments par page Ce décalage que nous appellerons OFFSET est égal à **INDICE*10** Une fois ce décalage fixé, la position horizontale d'affichage HOR peut être déterminée

Pour déterminer HOR, nous devons tenir compte de la largeur du rectangle de représentation (dans le cas présent la largeur du rectangle vaut 2 caractères) , et laisser un espace entre deux rectangles consécutifs Une donnée est donc représentée sur 3 caractères. L'affichage ne commence en fait qu'à la dixième colonne (les 10 premières colonnes sont utilisées pour contenir l'axe vertical, les graduations, etc...) Il faut donc ajouter 10 à chaque position horizontale d'affichage

HOR = ((I-OFFSET)*3)+10

Faisons varier I et calculons HOR Si I vaut 0 HOR vaut 10 Si I vaut 1 HOR vaut 13 Si I 25 HOR=25

La valeur maximale que pourra prendre HOR est 37

Comme pour les autres parties du programme, en fonction de l'indicateur ETAT, plusieurs sous-programmes devront être écrits

Représentation des valeurs positives et négatives

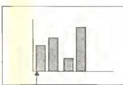
Les deux cas peuvent être traités dans le même sous-programme La différenciation se fera au niveau de la couleur de représentation rouge si les valeurs sont négatives, verte si les valeurs sont positives Dans ces deux cas, foutes les valeurs ont le même signe, ce qui signifie que ces valeurs seront représentées dans un même sens, vers le haut Deux lettres peuvent être affichées au pied de chaque rectangle de représentation Ces lettres permettent de différencier qualitativement les différentes valeurs.

Passons à l'ordinogramme de principe de ce sous-programme

En parcourant l'ordinogramme nous comprenons mieux le principe de l'affichage au pixel prés Les valeurs HIST,REEL et RESTE sont déterminées ainsi que GRAPHIC REEL est ensuite comparé avec la valeur 0 Si REEL vaut 0, cela signifie qu'il n'existe pas de partie entière, par contre rien ne nous permet d'affirmer qu'il n'existe pas de partie décimale Pour vérifier l'existence ou la non-existence d'une partie décimale, le sous-programme se branche un peu plus loin en aval Dans le cas ou REEL est non nul, une paire de

Programme valeur<10





Position de début d'affichage = 10

Les prochaines positions d'affichage sont 13 16 19 etc

POINTEUR	POSITION	POINTEUR	POSITION	POINTEUR	POSITION	POINTEUR	POSITION
0	10	10	10	20	10	30	10
1	13	11	13	21	13	31	13
2	16	12	16	22	18	32	16
3	19	13	19	23	19	33	19
4	22	14	22	24	22	34	22
5	25	15	25	25	28	35	25
6	28	16	28	26	28	36	28
7	31	17	31	27	31	37	31
8	34	18	34	28	34	38	34
9	37	19	37	29	37	39	37

caractères est affichée pour chaque valeur entière de REEL. Ainsi si REEL vaut 7, 7 paires de caractères sont affichées aux coordonnés XV et HOR. XV vaut au départ 19 et est décrémentée jusqu'à ce que le nombre de paires de caractères soit affiché.

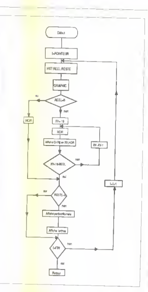
La valeur de RESTE est ensuite comparée avec 0. Si RESTE vaut 0, cela signifie qu'il n'existe pas de partie décimale. Si RESTE est non nul, la partie décimale est affichée à la suite de la partie entière. Les lettres chargées de différencier les différentes valeurs sont ensuite affichées.

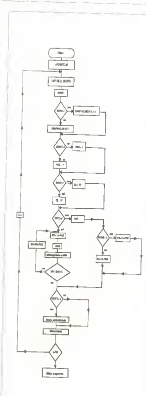
Affichage des positives ou négatives

Le problème est ici un peu plus complexe. En effet, les valeurs qui doivent être représentées ont des signes différents, ce qui signifie que les valeurs seront réparties de part et d'autre d'un axe médian. Pour réaliser cette représentation, nous aurons besoin de déterminer deux pas différents en fonction du signe de la valeur.

Si le signe est positif, le pas sera négatif, si le signe est négatif le pas sera positif. Cela peut paraître étrange et pourtant.

L'axe horizontal est un axe médian. Il est tracé au milieu de l'écran, soit environ à la dixième ligne. Voir figure. Les valeurs positives seront représentées vers le haut de l'écran en vert, les valeurs négatives seront représentées vers le bas





en rouge. La coordonnée verticale d'affichage XV peut voir sa valeur varier de 10 à 1 pour la représentation d'une valeur positive et de 10 à 19 pour la représentation d'une valeur négative. Le pas de la boucle doit être négatif pour la représentation positive et positif pour une représentation négative.
La couleur de représentation doit aussi être déterminée.
Une fois la prise en compte de ces différents paramètres, le reste de ce sous-programme est strictement identique à celui décrit ci-avant.

Conclusion
Ce programme peut s'adapter pour être intégré dans pratiquement tous les contextes: fichiers, tableaux, etc.

```
100 '###################################
110 '#          HISTOGRAMME             #
120 '###################################
130 CLS "Wob":CALL COLOR("WWb")
140 LOCATE (0,1):PRINT RPT$(" ",40)
150 '###################################
160 '#     CREATION DES DONNEES         #
170 '###################################
180 DIM TABLEAU$(100,2),PAGE$(100,2)
190 RANDOMIZE
200 FOR N=0 TO 9
210 X=INTRND(128)
220 IF X 65 THEN 210
230 IF X 65+26 THEN 210
240 Y=RND*2323!Y-INTRND(159)
250 NEGPOS=INTRND(100)
260 IF NEGPOS<=50 THEN Y=-Y ELSE Y=Y
270 TABLEAU$(N,0)=CHR$(X)&CHR$(X)
280 TABLEAU$(N,1)=STR$(Y)
290 NEXT
300 GOSUB 1950
310 '###################################
320 '*  INITIALISATION DES VARIABLES   #
330 '###################################
340 CODE=0:INDMAX=INT((N-1)/10):INDICE=0
350 POINTEUR=INDICE*10
360 FIN=POINTEUR+9
370 IF INDICE=INDMAX THEN FIN=N-1
380 '###################################
390 '*        DEBUT TRAITEMENT         #
400 '###################################
410 GOSUB 1070
420 ON ETAT GOSUB 1240,1240,1420
430 ON ETAT GOSUB 1770,1770,1830
440 CALL MESS1
450 RP$=KEY$
460 CLS "Wob":CALL COLOR("0Wb")
470 ON ETAT GOSUB 2210,2210,2300
480 ON ETAT GOSUB 2420,2420,2830
490 'PAUSE
500 IF INDICE 0 THEN CALL MESS2(CODE)
510 IF INDICE INDMAX THEN CALL MESS2(CODE)
520 IF CODE=1 THEN GOSUB 900:GOTO 560
530 GOSUB 640
540 ON ETAT GOSUB 2950,2950,3270
550  GOSUB 970
560 CALL MESS3
570 CALL MESS4(FACTEUR)
580 CALL MESS5(INDICE)
590 CODE=0
600 RP$=KEY$
610 IF ASC(RP$)=131 THEN GOSUB 720
620 IF ASC(RP$)=129 THEN GOSUB 810
630 GOTO 500
640 '###################################
650 '* TRANSFERT DANS UNE PAGE         #
660 '###################################
670 FOR I=POINTEUR TO FIN
680 PAGE$(I,0)=TABLEAU$(I,0)
690 PAGE$(I,1)=TABLEAU$(I,1)
700 NEXT I
710 RETURN
720 '###################################
730 '*  TRAITEMENT PAGE PRECEDENTE     #
740 '###################################
750 INDICE=INDICE-1
760 POINTEUR=INDICE*10
770 FIN=POINTEUR+9
780 IF INDICE=INDMAX THEN FIN=N-1
790 DRAPEAU=1
800 RETURN
810 '###################################
820 '*  TRAITEMENT PAGE SUIVANTE       #
830 '###################################
840 INDICE=INDICE+1
850 POINTEUR=INDICE*10
860 FIN=POINTEUR+9
870 IF INDICE=INDMAX THEN FIN=N-1
880 DRAPEAU=0
890 RETURN
900 '###################################
910 '#  DEBORDEMENT DE PAGE            #
920 '###################################
930 IF DRAPEAU=1 THEN INDICE=INDICE+1
940 IF DRAPEAU=0 THEN INDICE=INDICE-1
950 POINTEUR=INDICE*10
960 FIN=POINTEUR+10
970 IF INDICE=INDMAX THEN FIN=N-1
980 RETURN
990 '###################################
1000 '#      AFFICHE UNE PAGE          #
1010 '###################################
1020 CLS:CALL COLOR("0Wb"):LOCATE (5,1)
1030 FOR I=POINTEUR TO FIN
1040 PRINT I,PAGE$(I,0);" ";PAGE$(I,1)
1050 NEXT I
1060 RETURN
1070 '###################################
1080 '#    DETERMINATION DU SIGNE      #
1090 '###################################
1100 FOR I=0 TO N-1
1110 SIGNE=SGN(VAL(TABLEAU$(I,1)))
1120 IF SIGNE  1 THEN 1160
1130 NEXT
1140 ETAT=1
1150 RETURN
1160 FOR I=0 TO N-1
1170 SIGNE=SGN(VAL(TABLEAU$(I,1)))
1180 IF SIGNE<>-1 THEN 1220
1190 NEXT
1200 ETAT=2
1210 RETURN
1220 ETAT=3
1230 RETURN
1240 '###################################
1250 '#     DETERMINATION MAX/MIN      #
1260 '###################################
1270 MAX=VAL(TABLEAU$(0,1))
1280 FOR I=1 TO N-1
1290 IF VAL(TABLEAU$(I,1)) MAX THEN 1300 ELSE 1310
1300 MAX=VAL(TABLEAU$(I,1))
1310 NEXT
1320 MIN=VAL(TABLEAU$(0,1))
1330 FOR I=1 TO N-1
```

```
1340 IF VAL(TABLEAU$(I,1)) MIN THEN 1350 ELSE 1360
1350 MIN=VAL(TABLEAU$(I,1))
1360 NEXT
1370 MAXPOS=MAX:MINPOS=MIN
1380 IF ETAT=1 THEN 1410
1390 MAX=MINPOS
1400 MIN=MAXPOS
1410 RETURN
1420 '##################################
1430 '#     MAX MIN  POS ET NEV        #
1440 '##################################
1450 MAXPOS=VAL(TABLEAU$(0,1))
1460 FOR I=1 TO N-1
1470 IF VAL(TABLEAU$(I,1)) MAXPOS THEN 1480 ELSE 1490
1480 MAXPOS=VAL(TABLEAU$(I,1))
1490 NEXT
1500 RANG=0
1510 MINPOS=VAL(TABLEAU$(RANG,1))
1520 IF SGN(MINPOS)=-1 THEN RANG=RANG+1:GOTO 1510
1530 FOR I=RANG+1 TO N-1
1540 IF SGN(VAL(TABLEAU$(I,1)))=-1 THEN 1570
1550 IF VAL(TABLEAU$(I,1)) MINPOS THEN 1560 ELSE 1570
1560 MINPOS=VAL(TABLEAU$(I,1))
1570 NEXT
1580 RANG=0
1590 MAXNEV=VAL(TABLEAU$(RANG,1))
1600 IF SGN(MINNEV)=1 THEN RANG=RANG+1:GOTO 1590
1610 FOR I=RANG+1 TO N-1
1620 IF SGN(VAL(TABLEAU$(I,1)))=1 THEN 1650
1630 IF VAL(TABLEAU$(I,1))\MAXNEV THEN 1640 ELSE 1650
1640 MAXNEV=VAL(TABLEAU$(I,1))
1650 NEXT
1660 RANG=0
1670 MINNEV=VAL(TABLEAU$(RANG,1))
1680 IF SGN(MINNEV)=1 THEN RANG=RANG+1:GOTO 1670
1690 FOR I=RANG+1 TO N-1
1700 IF SGN(VAL(TABLEAU$(I,1)))=1 THEN 1730
1710 IF VAL(TABLEAU$(I,1)) MINNEV THEN 1720 ELSE 1730
1720 MINNEV=VAL(TABLEAU$(I,1))
1730 NEXT
1740 RETURN
1750 '##################################
1760 '#   DETERMINATTION ECH POSITIVE  #
1770 '##################################
1780 RAPPORT=MAX/MIN
1790 IF RAPPORT 170 THEN END
1800 ECH=ABS(17/MAX)
1810 IF ETAT=1 THEN C$="G" ELSE C$="R"
1820 RETURN
1830 '##################################
1840 '#   ECHELLE POSITIVE/NEGATIVE    #
1850 '##################################
1860 RAPPORTPOS=MAXPOS/MINPOS
1870 RAPPORTNEV=MAXNEV/MINNEV
1880 IF MAXPOS ABS(MAXNEV)THEN 1890 ELSE 1900
1890 DIV=MAXPOS:GOTO 1910
1900 DIV=-MAXNEV
1910 IF RAPPORTPOS 70 THEN END
1920 IF RAPPORTNEV 70 THEN END
1930 ECH=7/DIV
1940 RETURN
```

```
1950 '##################################
1960 '#   DEFINITION DES CARACTERES    #
1970 '##################################
1980 CALL CHAR(1,"0000000000000000FF")
1990 CALL CHAR(2,"000000000000FFFF")
2000 CALL CHAR(3,"0000000000FFFFFF")
2010 CALL CHAR(4,"00000000FFFFFFFF")
2020 CALL CHAR(5,"000000FFFFFFFFFF")
2030 CALL CHAR(6,"0000FFFFFFFFFFFF")
2040 CALL CHAR(7,"00FFFFFFFFFFFFFF")
2050 CALL CHAR(8,"000FFFFFFFFFFFFFFFF")
2060 CALL CHAR(9,"0FFFFFFFFFFFFFFFFFF")
2070 CALL CHAR(10,"FFFFFFFFFFFFFFFFFF")
2080 CALL CHAR(11,"FF00000000000000")
2090 CALL CHAR(12,"FFFF000000000000")
2100 CALL CHAR(13,"FFFFFF0000000000")
2110 CALL CHAR(14,"FFFFFFFF00000000")
2120 CALL CHAR(15,"FFFFFFFFFF000000")
2130 CALL CHAR(16,"FFFFFFFFFFFF0000")
2140 CALL CHAR(17,"FFFFFFFFFFFFFF00")
2150 CALL CHAR(18,"FFFFFFFFFFFFFFFF")
2160 CALL CHAR(19,"FFFFFFFFFFFFFFFF0")
2170 CALL CHAR(20,"FF[illegible]FF")
2180 CALL CHAR(21,"FF[illegible]")
2190 CALL CHAR(22,"[illegible]")
2200 RETURN
2210 '##################################
2220 '#      TRACE LES AXES            #
2230 '##################################
2240 CALL COLOR("1Wb")
2250 FOR X=19 TO 1 STEP -1
2260 LOCATE (X,3):PRINT CHR$(22)
2270 NEXT
2280 LOCATE (20,8):PRINT RPT$(CHR$(21),"2")
2290 RETURN
2300 '##################################
2310 '#   TRACE LES AXES POS NEV       #
2320 '##################################
2330 CALL COLOR("1Wb")
2340 FOR X=19 TO 10 STEP -1
2350 LOCATE (X,8):PRINT CHR$(22)
2360 NEXT
2370 LOCATE (X+1,8):PRINT RPT$(CHR$(20),22)
2380 FOR X=9 TO 1 STEP -1
2390 LOCATE (X,8):PRINT CHR$(22)
2400 NEXT
2410 RETURN
2420 '##################################
2430 '#        GRADUATION !            #
2440 '##################################
2450 Z=0
2460 IF ABS(MAX) 10 THEN 2650
2470 MAX$=STR$(INT(ABS(MAX))):L=LEN(MAX$):MULT=L-1
2480 IF L=2 THEN DICHO=1 ELSE DICHO=2
2490 RANG=VAL(SEG$(MAX$,DICHO,1))
2500 ENTIER=VAL(SEG$(MAX$,1,1))
2510 IF RANG=0 THEN RANGUT=RANG:GOTO 2550
2520 IF RANG 9 THEN 2530 ELSE 2540
2530 RANGUT=RANG+1::GOTO 2550
2540 RANGUT=1:ENTIER=ENTIER+1
2550 TOPUNI=ENTIER*10 MULT+(RANGUT*10 (MULT-1))
```

```
2560 UNI=[illegible] UNI 2
2570 FACTEUR=MULT
2580 IF ETAT=[illegible] THEN RETURN
2590 LOCATE (1,1):PRINT [illegible]
2600 LOCATE (19,1):PRINT UNI*1/10 MULT
2610 LOCATE ([illegible],1):PRINT UNI*2 10 MULT
2620 FACTEUR=MULT
2630 CALL MESSA(FACTEUR)
2640 [illegible]
2650 MAX=MAX+[illegible]
2660 IF ABS(MAX) [illegible] 1 THEN Z=Z+1:GOTO 2[illegible]5
2670 MAX$=STR$(ABS(MAX))
2680 RANG$=SEG$(MAX$,2,1)
2690 IF VAL(RANG$)=[illegible] THEN 27[illegible] ELSE 27[illegible]
2700 RANGLE=VAL(RANG$):GOTO 2720
2710 [illegible] +1
2720 ENTIER=[illegible](MAX$,1,1)
2730 [illegible]=[illegible](SEG$(MAX$,[illegible],1))*10+RANGLE
2740 [illegible]
2750 [illegible]
2760 FACTEUR=-1
2770 IF ETAT=[illegible] THEN RETURN
2780 LOCATE (19,[illegible]):PRINT
2790 LOCATE (1,1):PRINT UNI*1
2800 LOCATE ([illegible]):PRINT UNI*2
2810 CALL MESSA(FACTEUR)
2820 RETURN
2830 ####################################
2840 #       GRADUATION 2                 #
2850 !###################################
2860 IF RAPPORTPOS RAPPORTNEG THEN 2870 ELSE 2880
2870 MAX=[illegible]:GOTO 289[illegible]
2880 MAX=[illegible]NE
2890 GOSUB 24[illegible]
2900 LOCATE (1,1):PRINT
2910 LOCATE (19,1):PRINT UNI*2 /MULT
2920 LOCATE (1,1):PRINT ABS(UNI*2/10 MULT)
2930 CALL MESSA(FACTEUR)
2940 RETURN
2950 ####################################
2960 #        AFFICHAGE  PAGE 1           #
2970 !###################################
2980 CALL COLOR( wb )
2990 FOR X=1 TO 19:LOCATE (X,9):PRINT RPT$(" ",31
3000 NEXT
3010 LOCATE (21,1):PRINT RPT$(" ",40
3020 OFFSET=INDICE*10
3030 CALL COLOR("1"&C$)
3040 !       ***************************
3050 !       *CALCUL POSITIONS RELATIVES*
3060 !       ***************************
3070 FOR I=POINTEUR TO FIN
3080 HIST=ABS(VAL(PAGE$(1,I)))
3090 REEL=INT(HIST*ECH)
3100 RESTE=INT(10*(HIST*ECH-REEL))
3110 GRAPHIC=RESTE
3120 IF REEL=0 THEN 3[illegible] ELSE 3140
3130 HOR=((I-OFFSET)*3)+10
3140 FOR XV=19 TO 19-REEL STEP -1
3150 HOR=((I-OFFSET)*3)+10
3160 LOCATE (XV,HOR):PRINT CHR$(10)&CHR$(10)
3170 NEXT XV
3180 IF RESTE=0 THEN 3220
3190 CALL COLOR("1"&C$&"b")
3200 LOCATE (XV,HOR)
3210 PRINT CHR$(GRAPHIC)&CHR$(GRAPHIC)
3220 CALL COLOR("0"&C$&"B")
3230 LOCATE (21,HOR):PRINT SEG$(PAGE$(I,0),1,2)
3240 CALL COLOR("1"&C$)
3250 NEXT I
3260 RETURN
3270 ####################################
3280 '#   AFFICHE PAGE 2 VERSION 2       #
3290 ####################################
3300 CALL COLOR("1Wb")
3310 FOR X=1 TO 9:LOCATE (X,9):PRINT RPT$(" ",31)
3320 NEXT
3330 FOR X=11 TO 20:LOCATE (X,9):PRINT RPT$(" ",31
3340 NEXT
3350 LOCATE (21,1):PRINT RPT$(" ",40)
3360 OFFSET=INDICE*10
3370         ***************************
3380 !       *CALCUL POSITIONS RELATIVES*
3390         ***************************
3400 FOR I=POINTEUR TO FIN
3410 HIST=VAL(PAGE$(I,1))
3420 REEL=INT(HIST*ECH)
3430 RESTE=ABS(INT(10*(HIST*ECH-REEL)))
3440         ***************************
3450 !       * CHOIX DU CARACTERE      *
3460 !       ***************************
3470 SIGNE=SGN(HIST)
3480 IF SIGNE=1 THEN 3490 ELSE 3500
3490 GRAPHIC=RESTE:GOTO 3510
3500 GRAPHIC=RESTE+10
3510         ***************************
3520 !       * CHOIX COULEUR ET PAS     *
3530 !       ***************************
3540 IF SIGNE=1 THEN PAS=-1 ELSE PAS=1
3550 IF SIGNE=1 THEN C$="Y" ELSE C$="R"
3560         ***************************
3570 !       *CAS PARTICULIER           *
3580 !       *** ***********************
3590 IF REEL=0 THEN 3600 ELSE 3640
3600 HOR=((I-OFFSET)*3)+10
3610 CALL COLOR("1"&C$)
3620 IF SIGNE=1 THEN XV=10+PAS ELSE XV=10-PAS
3630 GOTO 3700
3640 FOR XV=10+PAS TO 10-REEL STEP PAS
3650 HOR=((I-OFFSET)*3)+10
3660 CALL COLOR("1"&C$)
3670 LOCATE (XV,HOR):PRINT CHR$(10)&CHR$(10)
3680 NEXT XV
3690 IF RESTE=0 THEN 3720
3700 LOCATE (XV,HOR)
3710 PRINT CHR$(GRAPHIC)&CHR$(GRAPHIC)
3720 CALL COLOR("0"&C$&"B")
3730 LOCATE (21,HOR):PRINT SEG$(PAGE$(I,0),1,2)
3740 CALL COLOR("1"&C$&"b")
3750 NEXT I
3760 RETURN
```

```
[illegible] '################################
[illegible] '#        MESSAGE 1             #
[illegible] '################################
[illegible] SUB MESS1
[illegible] CLS:CALL COLOR("1WLHI")
[illegible] LOCATE (10,14):PRINT "AAFPPPUUYYEEZZ"
[illegible] LOCATE (11,14):PRINT "AAFPPPUUYYEEZZ"
[illegible] C$="SSUUFF  UUNNEE  TTOOUUCCHHEE"
[illegible] LOCATE (14,7):PRINT C$
[illegible] LOCATE (15,7):PRINT C$
[illegible] SUBEND
[illegible] '################################
[illegible] '#        MESSAGE 2             #
[illegible] '################################
[illegible] SUB MESS2 [illegible]
[illegible] CALL COLOR [illegible] LOCATE [illegible]
[illegible] PRINT RPT$ [illegible],40)
[illegible] CALL COLOR [illegible] MIF [illegible]
[illegible] LOCATE [illegible]:PRINT "Commande interdite"
[illegible] [illegible]
[illegible] PAUSE [illegible]
[illegible] CALL COLOR [illegible]:LOCATE [illegible],1):PRINT RPT$ [illegible],40)
[illegible] SUBEND
[illegible] '################################
[illegible] '#   MESSAGE 3                  #
[illegible] '################################
[illegible] SUB MESS3
[illegible] CALL COLOR [illegible]
[illegible] LOCATE [illegible],1):PRINT [illegible]
[illegible] LOCATE [illegible]:PRINT [illegible]
[illegible] SUBEND
[illegible] '################################
[illegible] '#          MESSAGE 4           #
[illegible] '################################
[illegible] SUB MESS4(FACTEUR)
[illegible] CALL COLOR [illegible]:LOCATE [illegible]
[illegible] PRINT "Facteur=" [illegible] FACTEUR
[illegible] SUBEND
[illegible] '################################
[illegible] '#   MESSAGE 5                  #
[illegible] '################################
[illegible] SUB MESS5(INDICE)
[illegible] CALL COLOR [illegible]
[illegible] LOCATE [illegible]:PRINT "Page:";INDICE
[illegible] SUBEND
```

Comment utiliser de programme?

Ce programme trace des histogrammes à partir de valeurs aléatoires Ces valeurs aléatoires sont calculées et insérées dans un tableau à partir de la ligne 180 jusqu'à la ligne 290 Les lignes 250 et 260 peuvent être supprimées si l'on désire obtenir uniquement des valeurs positives ou négatives L'ordre de grandeur des valeurs peut être modifié en donnant une autre valeur à Y (ligne 240).

Le tableau peut contenir 100 valeurs et 100 libellés

Les différents sous-programmes

Lignes 1070 à 1230
Détermination du signe des valeurs

Lignes 1240 à 1410
Détermination du minima et du maxima pour une série de valeurs totalement positives ou totalement négatives

Lignes 1420 à 1740
Détermination des minimas et des maximas pour une série de valeurs positives ou négatives

Lignes 1750 à 1820
Détermination de l'échelle pour des valeurs totalement positives ou totalement négatives

Lignes 1830 à 1940
Détermination de l'échelle pour des valeurs positives ou négatives

Lignes 1950 à 2200
Sous-programme de redéfinition des caractères

Lignes 2210 à 2290
Trace les axes pour des valeurs totalement positives ou totalement négatives

Lignes 2300 à 2410
Trace les axes pour des valeurs positives ou négatives

Lignes 2420 à 2820
Calcule les graduations pour des valeurs totalement positives ou totalement négatives

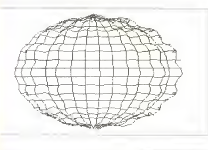
Lignes 2950 à 3260
Affiche des valeurs totalement positives ou totalement négatives

Lignes 3270 à 3760
Affiche des valeurs positives ou négatives

Lignes 3770 et plus
Sous-programmes chargés d'afficher les différents messages

```
100 '####################################
110 '#            SPHERE                #
120 '####################################
130 CALL HRON("W",1,20)
140 R=60
150 FOR T=0 TO PI STEP PI/20
160 F=-PI/2
170 GOSUB 360
180 X0=X+160:Y0=Y+100
190 FOR F=-PI/2 TO PI/2 STEP .1
200 GOSUB 360
210 CALL LINE("B",X0,Y0,X+160,Y+100)
220 X0=X+160:Y0=Y+100
230 NEXT
240 NEXT
250 FOR F=-PI/2 TO PI/2 STEP PI/20
260 T=0
270 GOSUB 360
280 X0=X+160:Y0=Y+100
290 FOR T=0 TO PI STEP .1
300 GOSUB 360
310 CALL LINE("B",X0,Y0,X+160,Y+100)
320 X0=X+160:Y0=Y+100
330 NEXT
340 NEXT
355 END
360 X=R*COS(F)*COS(T)*1.2
370 Y=R*SIN(F)
380 RETURN
```

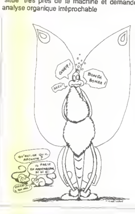
```
100 '####################################
110 '#      DEFORMATION                 #
120 '####################################
130 '
140 CALL HRON("B",1,20)
150 R=80
160 FOR T=.4 TO PI-.4 STEP PI/20
170 F=-PI/2
180 GOSUB 370
190 X0=X+160:Y0=Y+100
200 FOR F=-PI/2 TO PI/2 STEP .1
210 GOSUB 370
220 CALL LINE("G",X0,Y0,X+160,Y+100)
```



```
230 X0=X+160:Y0=Y+100
240 NEXT
250 NEXT
260 FOR F=-PI/2+.4 TO PI/2-.4 STEP PI/20
270 T=0
280 GOSUB 370
290 X0=X+160:Y0=Y+100
300 FOR T=0 TO PI STEP .1
310 GOSUB 370
320 CALL LINE("G",X0,Y0,X+160,Y+100)
330 X0=X+160:Y0=Y+100
340 NEXT
350 NEXT
360 END
370 A=COS(T*8):B=COS(F*8):C=COS(T*5):D=SIN(T*5)
380 H=R+20*ABS(A*B*C*D)
390 X=H*COS(F)*COS(T)*1.4
400 Y=H*SIN(F)
410 RETURN
```

Initiation à l'assembleur EXELMAX

L'assembleur est un langage de programmation très structuré mais très peu évolué. Concrétement, cela signifie qu'il est possible à l'aide d'Exelmax d'écrire un programme en développant de nombreuses petites routines disjointes qui seront utilisées tout au long du programme Ces routines correspondront au traitement d'une tâche très particulière. Le langage assembleur n'est pas un langage évolué: C'est-á-dire qu'il est très difficle sinon impossible d'aborder la programmation en assembleur comme vous le feriez pour un programme BASIC ou LOGO La programmation en assembleur se "situe" trés prés de la machine et demande une analyse organique irréprochable



La puissance du langage assembleur réside dans sa rapidité d'exécution et dans sa possibilité de traiter des cas trés particuliers et pointus.
Mais voilá, l'assembleur est un langage trés difficile à maîtriser En effet, une instruction CLS écrite en BASIC correspond à une vingtaine d'instructions assembleur D'autre part, ces instructions assembleur demande une très bonne connaissance du VDP (Video Display Processor). Devant l'aridité de ce langage, la plupart des programmeurs abandonne ce type de programmation. De toutes les manières, ce type de langage tend à disparaître de la panoplie du programmeur professionnel au profit du langage C. Le langage C est un hybride entre un langage évolué et un langage structuré
Il ne faut pas oublier que le développement d'un programme de quelques Koctets en assembleur peut prendre des mois Six instructions assembleur/heure est un très bon rythme
Nos chapitres consacrés á l'assembleur traiteront uniquement de l'écriture de routines assembleur utilisées conjointement avec le BASIC.

Ecriture dans le VDP

L'écriture d'un ou plusieurs octets dans le VDP est une opération fondamentale de la programmation assembleur qu'il faut parfaitement maîtriser Nous nous proposons á l'aide du programme assembleur EXELMAX d'écrire une routine qui affichera à l'écran un rectangle plein Les coordonnées de début d'affichage, le nombre de lignes, le nombre de colonnes ainsi que le code caractère et son attribut seront fournis à la routine assembleur par un programme BASIC. La routine assembleur sera implantée dans l'Exelmémoire à partir de l'adresse hexadécimale >8004 Pour la suite de l'exposé, une adresse précédée du signe > (supérieur) est toujours donnée en hexadécimal. Les valeurs données par le BASIC seront passées à la routine assembleur à partir de l'adresse >C7F0 (51184) qui correspond á la fin de la RAM système. La zone mémoire de >C7F0 à C7FF peut être utilisée pour passer les paramètres. Ces différents paramètres seront "pokés" à l'aide du sous-programme CALL POKE du BASIC.

Structure du VDP

Le VDP ou processeur vidéo dispose d'une RAM utilisateur de 64 Koctets pour un Exeltel et 32 Koctets pour un EXL 100. Sur les 64 Koctets de RAM, 2 Koctets sont utilisés en tant que RAM vidéo pour un écran texte (semi-graphique) et jusqu'à 24 Koctets sont utilisés en tant que RAM vidéo pour un écran haute résolution. La RAM VDP contient aussi les trois générateurs de caractères. L'emplacement de cette RAM vidéo est variable.
Néanmoins, lorsque le BASIC est utilisé, la RAM vidéo (semi-graphique) est placée en fin de RAM utilisateur. L'adresse de début de cette zone est contenue dans le registre double BAPPA et BAPPA-1 Ce double registre est situé dans la RAM système en >C103.

Registre POINTER sur 8 bits

MSb LSb

Registre COl et ROW

LSB MSB

Ce double registre contient l'adresse de la colonne et l adresse de la ligne

Registre STATUS

MSb LSb

Registres CM1 CM2 CM3 CM4

MSb LSb

Ces registres permettent de sélectionner les différents modes de travail et d affichage du VDP Ces registres sont initialisés par le BASIC

Registre BAPA

MSb LSb

Contient l adresse de début de la mémoire vidéo Cette adresse est également contenue dans le registre double BAPPA et BAPPA 1 dans la RAM système

Les registres BAGC0, BAGC1 BAGC2, BAGC3

MSb LSb

Ces registres contiennent l'adresse de chargement des différents générateurs de caractères Ces générateurs de caractères sont chargés en début de RAM VDP par le BASIC

Les registres internes du VDP

Un écran semi-graphique (texte) est composé de 80 octets par ligne + 2 octets pour le "border". Soit en tout 25 lignes de 82 octets, soit 2050 octets Pourquoi 80 octets ?
Pour afficher un caractère sur un écran texte, nous avons besoin de connaître le code caractère de 0 à 127, et son attribut Le code caractère est codé sur un octet, l'attribut est codé sur 1 octet et peut prendre les valeurs de 0 à 255. L'attribut fixe le numéro de générateur, la couleur du caractère, la couleur de fond, le mode vidéo, le mode double hauteur, double largeur etc..
Le VDP contient également des registres internes qui sont initialisés lorsque le BASIC est lancé Ces registres sont les suivants
3 générateurs de caractères sont chargés avec le BASIC BAGC3, BAGC0 et BAGC2 (reportez-vous au manuel de référence de l'Exelbasic+)
Comme nous n'utiliserons que des routines assembleur appelées par le BASIC, nous ne nous préoccuperons pas d'initialiser les registres du VDP, ni de charger les générateurs de caractères Par contre, nous pouvons étudier le registre de l'ATTRIBUT
L'affichage d'un caractère en mode texte demande deux octets. Un octet pour l'attribut, un octet pour le code de caractère



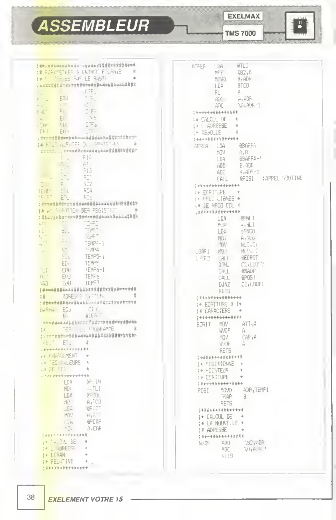
Programmation en assembleur

Nous utiliserons la cartouche EXELMAX et une Exelmémoire pour programmer la routine d'affichage de rectangles pleins
Pour les personnes qui débutent la programmation en langage d'assemblage, nous rappelons les manipulations nécessaires afin d'utiliser la cartouche Exelmax
Une fois la cartouche insérée, mettez sous tension l'ordinateur
Choisissez EDITOR (2) puis appuyez sur [ENVOI] deux fois consécutivement. Vous êtes maintenant sous l'éditeur d'Exelmax
Tapez le programme source donné ci-après
Une fois le programme correctement écrit, faites [CTL] et [C] Exelmax demande si vous désirez sauvegarder le programme source. Tapez Y puis donnez le nom du programme source et le périphérique sur lequel il sera stocké
C ECRAN SRC [ENVOI]
Le C indique que le programme source sera sauvegardé sur l'Exelmémoire. Une fois le programme source sauvegardé, ce dernier doit être assemblé Pour réaliser l'assemblage du programme source, choisissez l'option 3 du menu principal d'Exelmax

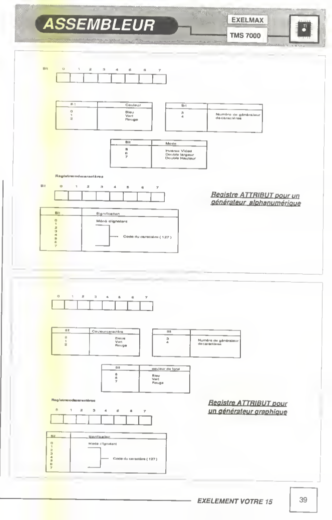
Exelmax demande:
Source file [VDP] ? [ENVOI]
Object file. [NUL] ? C:ECRAN OBJ [ENVOI]
Tapez ensuite deux fois [ENVOI]

Si vous désirez éditer un listage sur une imprimante, reportez-vous au manuel de référence de l'Exelmax.
Le programme source est assemblé Il nous reste maintenant à reloger ce programme objet à partir d'une certaine adresse Le programme assembleur peut tourner soit en RAM système soit en Exelmémoire Nous choisirons l'Exelmémoire pour des questions de simplicité En effet, une fois le programme objet relogé en début d'Exelmémoire (>8004), le programme BASIC n'aura qu'à charger le programme assembleur à l'aide d'un CALL LOAD à fournir les paramètres et exécuter le programme assembleur par un CALL DO. Donc, relogeons le programme assembleur en choisissant la quatrième option
Relocatable -> COM

Relocatable File Name: C:ECRAN.OBJ[ENVOI]
Command File NAME..:C:ECRAN.COM[ENVOI]
Run Time Address........:>8004 [ENVOI]



```
;################################
;# PARAMETRES D ENTREE FOURNIS   #
;# ENTREES PAR LE BASIC          #
;################################
NL        E         C7F1
          EQU       C7F2
          E         C7
NCO       EQ        C7F4
          EQU       C7F
CAP       EQU       C7F5
ATT       EQU       C7F
;################################
;# ATTRIBUTION DES REGISTRES     #
;################################
          E         R14
          EQU       R15
          EQU       R13
          EQU       R2
          E         R22
TEMP4     EQU       R24
TEMP6     EQU       R26
;################################
;# ATTRIBUTION DES REGISTRES     #
;################################
ATT       EQ        TEMP2
TLI       EQU       TEMP2-1
          EQU       TEMP3
          EQU       TEMP4-1
          EQ        TEMP4
          EQU       TEMP5-1
          EQU       TEMP5
TLI       EQU       TEMP6-1
TCO       EQU       TEMP6
NAD       EQU       TEMP7
;################################
;#        ADRESSE SYSTEME        #
;################################
BAFFA     EQU       C1
          B         @EEA
;################################
;#        DEBUT DU PROGRAMME     #
;################################
TPEL      EQU       $
;***************
;* CHARGEMENT  *
;* DES VALEURS *
;* PASSEES     *
;***************
          LDA       @FLIN
          MOV       A,TLI
          LDA       @FCOL
          MOV       A,TCO
          LDA       @FATT
          MOV       A,ATT
          LDA       @PCAP
          MOV       A,CAR
;***************
;* CALCUL DE   *
;* L ADRESSE   *
;* ECRAN       *
;* RELATIVE    *
;***************
ADFER     LDA       @TLI
          MPY       %82,A
          MOVD      B,ADR
          LDA       @TCO
          RL        A
          ADD       A,ADR
          ADC       %0,ADR-1
;***************
;* CALCUL DE   *
;* L ADRESSE   *
;* ABSOLUE     *
;***************
ADREA     LDA       @BAFFA
          MOV       A,B
          LDA       @BAFFA-1
          ADD       B,ADR
          ADC       A,ADR-1
          CALL      @POSI     ;APPEL SOUTINE
;***************
;* ECRITURE    *
;* NLI LIGNES  *
;* DE NCO COL  *
;***************
          LDA       @PNLI
          MOV       A,NLI
          LDA       @PNCO
          MOV       A,NCO
          MOV       NLI,C1
LOOP1     MOV       NCO,C2
LOOP2     CALL      @ECRIT
          DJNZ      C2,LOOP2
          CALL      @NADR
          CALL      @POSI
          DJNZ      C1,LOOP1
          RETS
;***************
;* ECRITURE D 1*
;* CARACTERE   *
;***************
ECRIT     MOV       ATT,A
          MVDF      A
          MOV       CAR,A
          MVDF      A
          RETS
;***************
;* POSITIONNE  *
;* POINTEUR    *
;* ECRITURE    *
;***************
POSI      MOVD      ADR,TEMP1
          TRAP      9
          RETS
;***************
;* CALCUL DE   *
;* LA NOUVELLE *
;* ADRESSE     *
;***************
NADR      ADD       %82,ADR
          ADC       %0,ADR-1
          RETS
```


ASSEMBLEUR
EXELMAX
TMS 7000
Bit
0 1 2 3 4 5 6 7
Bit
Couleur
0 1 2
Bleu Vert Rouge
Bit
3 4
Numéro de générateur de caractères
Bit
Mode
5 6 7
Inverse Video
Double largeur
Double Hauteur
Registre de caractères
Bit
0 1 2 3 4 5 6 7
Bit
Signification
0 1 2 3 4 5 6 7
Mode clignotant
Code du caractère (127)
Registre ATTRIBUT pour un générateur alphanumérique
0 1 2 3 4 5 6 7
Bit
Couleur caractère
0 1 2
Bleu Vert Rouge
Bit
3 4
Numéro de générateur de caractères
Bit
couleur de fond
5 6 7
Bleu Vert Rouge
Registre de caractères
0 1 2 3 4 5 6 7
Bit
Signification
0 1 2 3 4 5 6 7
Mode clignotant
Code du caractère (127)
Registre ATTRIBUT pour un générateur graphique

Nous avons maintenant dans l'Exelmémoire 3 programmes
ECRAN SRC
ECRAN OBJ
ECRAN COM

Structure du programme source

Avant de programmer la moindre ligne en assembleur, il est indispensable de réaliser un ordinogramme de principe.

Les différents blocs du programme:

Paramètres d'entrée fournis et testés par le BASIC
Ces paramètres seront introduits à l'aide du sous-programme intégré CALL POKE. La zone de transmission de ces paramètres est située en fin de RAM système en >C7F1 soit 51185.

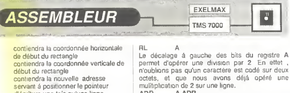
PLIN	représente la coordonnée horizontale de début du rectangle
PCOL	représente la coordonnée verticale de début du rectangle
PLIN	représente le nombre de lignes du rectangle
PCOL	représente le nombre de colonnes du rectangle
PATT	représente la valeur de l'attribut
PCAR	représente le code du caractère constitutif du rectangle

Equivalences des registres
Dans ce bloc de programme, on définit les registres qui seront utilisés. 127 registres sont disponibles avec le 7040. Attention, avec une utilisation BASIC, on ne peut utiliser que les registres R14 à 24 R24 est le <u>maximum.</u> En effet le BASIC utilise une grande partie des registres disponibles.

Attribution des registres
Dans ce bloc de programme, on attribue aux registres définis antérieurement des labels.

ADR	contiendra l'adresse servant à positionner le pointeur d'écriture en RAM vidéo
NLI	contiendra le nombre de lignes du rectangle
NCO	contiendra le nombre de colonnes du rectangle
ATT	contiendra la valeur de l'attribut
CAR	contiendra le code caractère
C1	identique à NLI
C2	identique à NCO

Début
Valeurs pokées
Equivalences des registres
Attribution des registres
Chargement des valeurs pokées
Calcul de l'adresse relative
Calcul de l'adresse absolue
C1=NLI
C2=NCO
Appel routine ECRIT
C2=C2 1
C2=0
Non
Appel routine NADR
Appel routine POSI
C1=C1 1
C1=0
Non



TLI contiendra la coordonnée horizontale de début du rectangle
TCO contiendra la coordonnée verticale de début du rectangle
NAD contiendra la nouvelle adresse servant à positionner le pointeur d'écriture une fois qu'une ligne entière est écrite

Adresse système

BAPPA Contient l'adresse de début d'écran texte

Le BR (branch) @DEBUT branche le programme à son commencement réel Tout ce que l'on a écrit n'a qu'une valeur déclarative

Chargement des valeurs pokées
Ce bloc de programme a pour tâche de récupérer les valeurs introduites à partir du BASIC pour être traitées par le programme assembleur.
LDA @PLIN
Le registre A est chargé avec la valeur contenue dans PLIN (coordonnée horizontale)
MOV A,TLI
Le contenu de A est transféré dans le registre TLIN (registre declaré TEMP6-1)
On répète l'opération pour la coordonnée verticale, la valeur de l'attribut et le code caractère. A la fin de ce bloc de programme, nous avons écrit les registres
TLI, TCO, ATT et CAR.

Calcul de l'adresse écran relative
Ce bloc de programme est chargé de calculer l'adresse relative, c'est-à-dire l'emplacement virtuel du début du rectangle
LDA @TLI
Le registre A est chargé avec la valeur de la coordonnée horizontale du rectangle
MPY %82,A
Sachant qu'une l"igne texte" est constitué de 82 octets (80 octets pour 40 caractères + 2 octets pour le "border"), on multiplie le contenu du registre A par 82
MOVD B,ADR
La valeur obtenue en multipliant le contenu du registre A par une quantité ou le contenu d'un autre registre est toujours placée dans B et A. On transfère sur deux octets le contenu du registre A et B dans le registre déclaré ADR (2octets ADR et ADR-1 -)
LDA @TCO
Le registre A est à nouveau chargé avec la coordonnée verticale du rectangle

RL A
Le décalage à gauche des bits du registre A permet d'opérer une division par 2 En effet , n'oublions pas qu'un caractère est codé sur deux octets, et que nous avons déjà opéré une multiplication de 2 sur une ligne.
ADD A,ADR
On additionne le résultat dans ADR et on ajoute la retenue s'il y a lieu
ADC %0,ADR-1

Calcul de l'adresse absolue
Ce bloc de programme est chargé d'additionner la valeur contenue dans BAPPA (début de la RAM vidéo) à la valeur contenue dans ADR
LDA @BAPPA
Le registre A est chargé avce le contenu de BAPPA
MOV A,B
Le contenu du registre A est tranféré dans B
LDA A,BAPPA-1
Le registre A est chargé avec le contenu de BAPPA-1
ADD B,ADR
On additionne le contenu de B à la quantité contenue dans ADR
ADC A,ADR-1
On additionne le contenu de A à la quantité contenue dans ADR-1 et on ajoute la retenue s'il y a lieu.
CALL @POSI
Cette ligne d'instructions appelle un bloc de programme (tout comme un sous-programme en BASIC) Ce sous-programme est chargé de positionner le pointeur d'écriture dans la RAM vidéo

Ecriture de NPLI lignes de NPCO colonnes

```
LDA    @PNLI
MOV    A,NLI
LDA    @PNCO
MOV    A,NCO
```

Ce bloc de lignes prend les valeurs pokées et les range dans des registres. Ces registres contiennent le nombre de lignes et le nombre de colonnes Le contenu de ces registres est ensuite transféré dans des registres temporaires servant de compteurs de boucles

```
        MOV    NLI,C1
LOOP1   MOV    NCO,C2
LOOP2   CALL   @ECRIT
        DJNZ   C2,LOOP2
        CALL   @NADR
        CALL   @POSI
        DJNZ   C1,LOOP1
```



Ce bloc des lignes fonctionne de la manière suivante. Le premier compteur de boucles est chargé avec le nombre de lignes du rectangle. Le second compteur de boucles est chargé avec le nombre de colonnes du rectangle. On appelle la routine d'écriture, on décrémente le compteur des colonnes tant que celui-ci n'est pas égal à zéro. Quand le compteur des colonnes est égal à zéro, une ligne est écrite. On calcule la nouvelle adresse de positionnement du pointeur d'écriture (CALL @NADR), on positionne le pointeur d'écriture et on recommence la procédure d'écriture tant que le compteur de lignes n'est pas égal à zéro.

Ecriture d'un caractère
Ce bloc de programme écrit un caractère dans la RAM vidéo.

```
ECRIT   MOV     ATT,A
        WVDP A
        MOV     CAR,A
        WVDP A
        RETS
```

Le registre A est chargé avec le contenu du registre ATT (contenant la valeur de l'attribut). L'instruction WVDP écrit la valeur contenue dans le registre A. La même opération est reproduite pour le code caractère

Positionne pointeur d'écriture
Ce bloc de programme positionne le pointeur d'écriture dans la RAM vidéo.

```
POSI    MOVD    ADR,TEMP1
        TRAP    8
        RETS
```

Le contenu du registre double ADR et ADR-1 est transféré dans le registre TEMP1. L'instruction TRAP 8 positionne le pointeur sur l'adresse contenue dans TEMP1 et TEMP1-1.

Calcul de la nouvelle adresse d'écriture
Ce bloc de programme tranfère dans ADR et ADR-1 la nouvelle adresse d'écriture. Pour passer à une autre ligne d'écriture, il suffit d'ajouter 82 à la valeur précédente

```
NADR    ADD     %82,ADR
        ADC     %0,ADR-1
        RETS
```

Le programme BASIC

Nous avons écrit une routine en assembleur que nous allons utiliser au sein d'un programme BASIC. Le programme BASIC que nous proposons utilise la routine assembleur pour afficher des rectangles pleins de différentes couleurs.

```
100 '###################################
110 '#    CADRE EN ASSEMBLEUR          #
120 '###################################
130 CLS "6BB":CALL COLOR("06B )
140 CALL LOAD("ECRAN")
150 A=56:C=12
160 X=0:Y=0:Z=24:W=40
170 IF X 10 THEN 230
180 CALL PARA(X,Y,Z,W,A,C)
190 CALL DO("ECRAN")
200 X=X+1:Y=Y+1:Z=Z-2:W=W-2:A=A+1
210 PAUSE .5
220 GOTO 170
230 A=0:C=12
240 IF X=0 THEN 150
250 X=X-1:Y=Y-1:Z=Z+2:W=W+2
260 CALL PARA(X,Y,Z,W,A,C)
270 CALL DO("ECRAN"):PAUSE .5
280 GOTO 240
290 SUB PARA(Y,Y,Z,W,A,C)
300 CALL POKE(51185,X,Y,Z,W,A,C)
310 SUBEND
```

SOMMAIRE

Directeur de la publication EXELVISION
Rubriques initiation Patrice CHAILLAN
Rubrique telematique Nicolas BELLOIR